G 16

```
FFFFFFFFFFFFFFF  DDDDDDDDDDDD      LLL
FFFFFFFFFFFFFFF  DDDDDDDDDDDD      LLL
FFFFFFFFFFFFFFF  DDDDDDDDDDDD      LLL
FFF              DDD         DDD  LLL
FFF              DDD         DDD  LLL
FFF              DDD         DDD  LLL
FFF              DDD         DDD  LLL
FFF              DDD         DDD  LLL
FFF              DDD         DDD  LLL
FFFFFFFFFFFF     DDD         DDD  LLL
FFFFFFFFFFFF     DDD         DDD  LLL
FFFFFFFFFFFF     DDD         DDD  LLL
FFF              DDD         DDD  LLL
FFF              DDD         DDD  LLL
FFF              DDD         DDD  LLL
FFF              DDD         DDD  LLL
FFF              DDD         DDD  LLL
FFF              DDD         DDD  LLL
FFF              DDDDDDDDDDDD     LLLLLLLLLLLLLLL
FFF              DDDDDDDDDDDD     LLLLLLLLLLLLLLL
FFF              DDDDDDDDDDDD     LLLLLLLLLLLLLLL
```

FILEID**FDLDRIVER

K 8

```
FFFFFFFFFF  DDDDDDDD    LL          DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
FFFFFFFFFF  DDDDDDDD    LL          DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
FF          DD      DD  LL          DD      DD  RR      RR      II      VV      VV  EE          RR      RR
FF          DD      DD  LL          DD      DD  RR      RR      II      VV      VV  EE          RR      RR
FF          DD      DD  LL          DD      DD  RR      RR      II      VV      VV  EE          RR      RR
FF          DD      DD  LL          DD      DD  RR      RR      II      VV      VV  EE          RR      RR
FFFFFFFF    DD      DD  LL          DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
FFFFFFFF    DD      DD  LL          DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
FF          DD      DD  LL          DD      DD  RR  RR          II      VV      VV  EE          RR  RR
FF          DD      DD  LL          DD      DD  RR  RR          II      VV      VV  EE          RR  RR
FF          DD      DD  LL          DD      DD  RR    RR        II        VV  VV    EE          RR    RR
FF          DD      DD  LL          DD      DD  RR    RR        II        VV  VV    EE          RR    RR      ....
FF          DDDDDDDD    LLLLLLLLLL  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR    ....
FF          DDDDDDDD    LLLLLLLLLL  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR    ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 %TITLE  'FDLDRIVER'
0 %SBTTL  'FDL Parse Table Drivers'
0 MODULE  FDLDRIVER          ( IDENT='V04-000',
0                            ADDRESSING_MODE ( EXTERNAL = GENERAL ),
0                            ADDRESSING_MODE ( NONEXTERNAL = GENERAL ),
0                            OPTLEVEL=3
0                            ) =
1 BEGIN
1
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! Facility:      RMS-32 FDL Utilities
1 !
1 ! Abstract:
1 !
1 ! Contents:
1 !               GET_LINE
1 !               UPCASE
1 !               SET_LINE
1 !               SET_TERM
1 !               SET_PRIMARY
1 !               SET_SECONDARY
1 !               START_STR
1 !               END_STR
1 !               SET_DATE_TIME
1 !               SET_COMMENT
1 !               SYNTAX_ERROR
1 !               ERROR_CHK
1 !               NEGATE
1 !               SET_BLANK
1 !               CLR_BLANK
1 !               FDL$$READ_ERROR
1 !               RMS_ERROR
1 !               RMS_OPEN_ERROR
```

```
;      0058  1 !
;      0059  1 ! Environment:
;      0060  1 !
;      0061  1 !              VAX/VMS Operating System
;      0062  1 !
;      0063  1 !--
```

```
     1 !
     1 ! Author:        Keith B Thompson        Creation date:  January-1981
     1 !
     1 ! Modified by:
     1 !
     1 !        V03-012 KFH0009         Ken Henderson           23 Aug 1983
     1 !                Fix to FDL$$GET_LINE to allow null
     1 !                FDL spec string.
     1 !                Fix calls to GET_VM and FREE_VM.
     1 !
     1 !        V03-011 KFH0008         Ken Henderson           10 Aug 1983
     1 !                Fixes to END_STR and SET_DATE_TIME
     1 !                Addition of EXTRACT_QUOTE routine
     1 !                Addition of TRIM_LEADING routine
     1 !
     1 !        V03-010 KFH0007         Ken Henderson           29 Jul 1983
     1 !                Check status of LIB$TPARSE call
     1 !
     1 !        V03-009 KFH0006         Ken Henderson           26 Apr 1983
     1 !                Fixed call to $BINTIM
     1 !
     1 !        V03-008 KFH0005         Ken Henderson           30 Dec 1982
     1 !                Fixed broken branches
     1 !
     1 !        V03-007 KFH0004         Ken Henderson           21 Dec 1982
     1 !                Fixed signal of FDL$_UNSECKW
     1 !
     1 !        V03-006 KFH0003         Ken Henderson           15-Nov-1982
     1 !                Added support for default and
     1 !                main parses
     1 !                Added support for more than 32
     1 !                secondaries per primary
     1 !                Added move to .FDL$GL_STNUMPTR of
     1 !                .FDL$GL_STMNTNUM
     1 !
     1 !        V03-005 KFH0002         Ken Henderson           6-Oct-1982
     1 !                Removed numtype
     1 !                Added support for ACL primary
     1 !                Added support for FDL STRINGS
     1 !
     1 !        V03-004 KFH0001         Ken F. Henderson        28-Jul-1982
     1 !                Changed terminator character from "/" to "\"
     1 !
     1 !        V03-003 KBT0067         Keith B. Thompson       23-Jun-1982
     1 !                Add support for multiple keywords per line
     1 !
     1 !        V03-002 KBT0029         Keith Thompson          30-Mar-1982
     1 !                Add upcase/lowercase processing and date/time routine
     1 !
     1 !        V03-001 KBT0019         Keith Thompson          22-Mar-1982
     1 !                Fix error message processing
     1 !
     1 !****
```

```
1
1 PSECT
1               OWN     = _FDL$OWN          (PIC),
1               GLOBAL  = _FDL$GLOBAL       (PIC),
1               PLIT    = _FDL$PLIT         (SHARE,PIC),
1               CODE    = _FDL$CODE         (SHARE,PIC);
1
1 LIBRARY 'SYS$LIBRARY:STARLET';
1 REQUIRE 'SRC$:FDLUTIL';
1 REQUIRE 'LIB$:FDLPARDEF';
1
1 EXTERNAL ROUTINE
1               FDL$$GET_VM,
1               FDL$$FREE_VM,
1               LIB$TPARSE,
1               STR$TRIM,
1               SYS$BINTIM;
1
1 DEFINE_ERROR_CODES;
1
1 FORWARD ROUTINE
1               EXTRACT_QUOTE,
1               TRIM_LEADING,
1               UPCASE                  : NOVALUE,
1               FDL$$SET_PRIMARY,
1               FDL$$SET_SECONDARY,
1               FDL$$START_STR,
1               FDL$$END_STR,
1               FDL$$SET_COMMENT,
1               FDL$$SYNTAX_ERROR,
1               FDL$$ERROR_CHK,
1               FDL$$READ_ERROR         : NOVALUE;
1
1 EXTERNAL
1
1 !             Parse control
1 !
1               FDL$AB_LINE             : DESC_BLK,
1               FDL$AB_UPCASED          : DESC_BLK,
1               FDL$AB_ITEM             : DESC_BLK,
1               FDL$AB_FDL_STRING       : DESC_BLK,
1               FDL$AB_PRE_PARSE_BLOCK  : BLOCK [ ,BYTE ],
1               FDL$AB_PRE_PARSE_STATE,
1               FDL$AB_PRE_PARSE_KEY,
1               FDL$GL_STNUMPTR,
1               FDL$GL_MAXLINE,
1               FDL$AB_CTRL             : BLOCK [ ,BYTE ],
1               FDL$GL_PRIMARY,
1               FDL$GL_PRINUM,
1               FDL$AB_PRICTRL          : BLOCK [ ,BYTE ],
1               FDL$GL_SECONDARY,
1               FDL$GL_SECNUM,
1               FDL$AB_SECCTRL          : BITVECTOR [ FDL$K_SCTRL_VEC ],
1               FDL$AB_SECCTRLL         : VECTOR [ FDL$K_SCTRL_LONG, LONG ],
1               FDL$GL_QUALIFIER,
1               FDL$GL_NUMBER,
1               FDL$GL_SWITCH,
;  177     0897  1         FDL$GL_PROTECTION,
;  178     0898  1         FDL$AL_DATE_TIME        : VECTOR [ ,LONG ],
;  179     0899  1         FDL$AB_STRING           : DESC_BLK,
;  180     0900  1         FDL$AB_COMMENT          : DESC_BLK,
;  181     0901  1         FDL$GL_STMNTNUM,
;  182     0902  1         FDL$AB_FDL_RAB          : $RAB_DECL;
;  183     0903  1
;  184     0904  1 LITERAL
;  185     0905  1         SMALL_A         = 97,           ! ASCII character "a"
;  186     0906  1         SMALL_Z         = 122,          ! ASCII character "z"
;  187     0907  1         UPCASE_MASK     = 32,           ! Mask to convert to upercase ASCII
;  188     0908  1         COMMENT_MARK    = 33;           ! ASCII character "!"
;  189     0909  1
;  190     0910  1 OWN
;  191     0911  1         STRING_DESC     : DESC_BLK;
```

```
1 %SBTTL  'GET_LINE'
1 GLOBAL ROUTINE  FDL$$GET_LINE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Set up a new item for the parse tables.  If there are no
1 !       more items on a line it then reads from the input file (or uses
1 !       the FDL STRING)
1 !       It then upcases it; inits some values and returns
1 !
1 ! Calling Sequence:
1 !
1 !       Called from the parse tables
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     TPARSE_ARGS;
2
2     ! Main processing loop
2     !
2     DO
3         BEGIN
3
3         ! If there are no more items in the line get a new line
3         !
3         IF .FDL$AB_ITEM [ DSC$W_LENGTH ] EQL 0
3         THEN
4             BEGIN
4
4             IF .FDL$AB_CTRL [ FDL$V_STRING_SPEC ]
4             THEN
5                 BEGIN
5
5                 ! Only go thru once for the string.
5                 ! Don't go thru at all if the string is null.
5                 !
```

```
6                  IF (
7                  (.FDL$AB_CTRL [ FDL$V_USED_STRING ])
6                  OR
7                  (.FDL$AB_FDL_STRING [ DSC$W_LENGTH ] EQLU 0)
5                  ) THEN
5                      RETURN 0;
5
5                  CH$MOVE ( .FDL$AB_FDL_STRING [ DSC$W_LENGTH ],
5                            .FDL$AB_FDL_STRING [ DSC$A_POINTER ],
5                            .FDL$AB_LINE [ DSC$A_POINTER ] );
5
5                  FDL$AB_LINE [ DSC$W_LENGTH ] = .FDL$AB_FDL_STRING [ DSC$W_LENGTH ];
5                  FDL$AB_CTRL [ FDL$V_USED_STRING ] = _SET;
5
5                  END
4              ELSE
5                  BEGIN
5
5                  ! Loop until we get a non-zero line
5                  !
5                  DO
6                      BEGIN
6
6                      ! Get the new line from the FDL file.
6                      !
6                      RET_ON_ERROR( $GET ( RAB=FDL$AB_FDL_RAB,ERR=FDL$$READ_ERROR ) );
6
6                      END
6
6                  UNTIL ( FDL$AB_LINE [ DSC$W_LENGTH ] =
5                                          .FDL$AB_FDL_RAB [ RAB$W_RSZ ] ) NEQ 0;
5
4                  END;
4
4              ! Up case the whole line and move it into the upcase buffer
4              !
4              UPCASE();
4
4              ! Point the tables to the upcased line
4              !
4              FDL$AB_ITEM [ DSC$A_POINTER ] = .FDL$AB_UPCASED [ DSC$A_POINTER ]
4
3              END;
3
3          ! Point to the next item
3          !
3          FDL$AB_ITEM [ DSC$A_POINTER ] = .FDL$AB_ITEM [ DSC$A_POINTER ] +
3                                                  .FDL$AB_ITEM [ DSC$W_LENGTH ];
3
4          BEGIN
4
4          ! Get the string
4          !
4          FDL$AB_PRE_PARSE_BLOCK [ TPA$L_STRINGCNT ] =
4                                      .FDL$AB_UPCASED [ DSC$W_LENGTH ]
5                                      - (
5                                          .FDL$AB_ITEM [ DSC$A_POINTER ]
```

```
                                                - .FDL$AB_UPCASED [ DSC$A_POINTER ]
                                          );
            FDL$AB_PRE_PARSE_BLOCK [ TPA$L_STRINGPTR ] = .FDL$AB_ITEM [ DSC$A_POINTER ];

            ! Find where to chop it off - the Tparse will set these flags if
            ! it finds " or '
            !
            FDL$AB_CTRL [ FDL$V_QUOTE_PRES ] = _CLEAR;
            FDL$AB_CTRL [ FDL$V_APOST_PRES ] = _CLEAR;

            RET_ON_ERROR ( LIB$TPARSE (
            FDL$AB_PRE_PARSE_BLOCK,FDL$AB_PRE_PARSE_STATE,FDL$AB_PRE_PARSE_KEY ));

            ! Now set up the main tparse block to find our 'line'
            !
            TPARSE_BLOCK [ TPA$L_STRINGPTR ] = .FDL$AB_ITEM [ DSC$A_POINTER ];
            TPARSE_BLOCK [ TPA$L_STRINGCNT ] =
                        .FDL$AB_PRE_PARSE_BLOCK [ TPA$L_STRINGPTR ]
                        - .FDL$AB_ITEM [ DSC$A_POINTER ];

            FDL$AB_ITEM [ DSC$W_LENGTH ] = .TPARSE_BLOCK [ TPA$L_STRINGCNT ];

            END         ! of local

            END         ! of main loop

        UNTIL .FDL$AB_ITEM [ DSC$W_LENGTH ] NEQ 0;

        FDL$GL_STMNTNUM = .FDL$GL_STMNTNUM + 1;

        ! Update the user's cell that contains the statement number.
        !
        IF .FDL$AB_CTRL [ FDL$V_STVALID ]
        THEN
            .FDL$GL_STNUMPTR = .FDL$GL_STMNTNUM;

        ! Since there is a new secondary for each item clear some flags
        !
        FDL$GL_SECONDARY    = _CLEAR;
        FDL$GL_SECNUM       = _CLEAR;
        FDL$GL_SWITCH       = _CLEAR;
        FDL$GL_PROTECTION   = _CLEAR;
        FDL$AB_STRING [ DSC$W_LENGTH ] = 0;
        FDL$AB_CTRL [ FDL$V_WARNING ] = _CLEAR;
        FDL$AB_CTRL [ FDL$V_COMMENT ] = _CLEAR;
        FDL$AB_CTRL [ FDL$V_LINECMT ] = _CLEAR;

        RETURN SS$_NORMAL

        END;

                                                        .TITLE  FDLDRIVER VAX-11 FDL Utilities
                                                        .IDENT  \V04-000\

                                                        .PSECT  _FDL$OWN,NOEXE,  PIC,2
                                                00000 STRING_DESC:
                                                              .BLKB    8

                                                              .EXTRN   FDL$$GET_VM, FDL$$FREE_VM
                                                              .EXTRN   LIB$TPARSE, STR$TRIM
                                                              .EXTRN   SYS$BINTIM, FDL$_FACILITY
                                                              .EXTRN   FDL$_FAO_MAX, FDL$_ABKW
                                                              .EXTRN   FDL$_ABPRIKW, FDL$_CREATE
                                                              .EXTRN   FDL$_CREATED, FDL$_CREATEDSTM
                                                              .EXTRN   FDL$_FDLERROR, FDL$_ILL_ARG
                                                              .EXTRN   FDL$_INSVIRMEM, FDL$_INVBLK
                                                              .EXTRN   FDL$_INVDATIM, FDL$_MULPRI
                                                              .EXTRN   FDL$_MULSEC, FDL$_NOQUAL
                                                              .EXTRN   FDL$_NULLPRI, FDL$_OPENFDL
                                                              .EXTRN   FDL$_OUTORDER, FDL$_OPENOUT
                                                              .EXTRN   FDL$_WRITEERR, FDL$_READERR
                                                              .EXTRN   FDL$_RFLOC, FDL$_TITLE
                                                              .EXTRN   FDL$_SYNTAX, FDL$_VALPRI
                                                              .EXTRN   FDL$_UNQUAKW, FDL$_UNPRIKW
                                                              .EXTRN   FDL$_UNSECKW, FDL$_WARNING
                                                              .EXTRN   FDL$AB_LINE, FDL$AB_UPCASED
                                                              .EXTRN   FDL$AB_ITEM, FDL$AB_FDL_STRING
                                                              .EXTRN   FDL$AB_PRE_PARSE_BLOCK
                                                              .EXTRN   FDL$AB_PRE_PARSE_STATE
                                                              .EXTRN   FDL$AB_PRE_PARSE_KEY
                                                              .EXTRN   FDL$GL_STNUMPTR
                                                              .EXTRN   FDL$GL_MAXLINE, FDL$AB_CTRL
                                                              .EXTRN   FDL$GL_PRIMARY, FDL$GL_PRINUM
                                                              .EXTRN   FDL$AB_PRICTRL, FDL$GL_SECONDARY
                                                              .EXTRN   FDL$GL_SECNUM, FDL$AB_SECCTRL
                                                              .EXTRN   FDL$AB_SECCTRLL
                                                              .EXTRN   FDL$GL_QUALIFIER
                                                              .EXTRN   FDL$GL_NUMBER, FDL$GL_SWITCH
                                                              .EXTRN   FDL$GL_PROTECTION
                                                              .EXTRN   FDL$AL_DATE_TIME
                                                              .EXTRN   FDL$AB_STRING, FDL$AB_COMMENT
                                                              .EXTRN   FDL$GL_STMNTNUM
                                                              .EXTRN   FDL$AB_FDL_RAB, SYS$GET

                                                              .PSECT   _FDL$CODE,NOWRT,  SHR,  PIC,2

                                          0FFC 00000          .ENTRY   FDL$$GET_LINE, Save R2,R3,R4,R5,R6,R7,R8,-   ; 0913
                                                                       R9,R10,R11                                   ;
                     5B 00000000G   00   9E 00002          MOVAB    FDL$AB_FDL_STRING, R11                       ;
                     5A 00000000G   00   9E 00009          MOVAB    FDL$AB_PRE_PARSE_BLOCK+12, R10               ;
                     59 00000000G   00   9E 00010          MOVAB    FDL$AB_CTRL, R9                              ;
                     58 00000000G   00   9E 00017          MOVAB    FDL$AB_ITEM, R8                              ;
                     57             68   3C 0001E          MOVZWL   FDL$AB_ITEM, R7                              ; 0958
                                    62   12 00021          BNEQ     6$                                           ;
     2B        01    A9             04   E1 00023 1$:      BBC      #4, FDL$AB_CTRL+1, 4$                        ; 0962
     03        01    A9             05   E1 00028          BBC      #5, FDL$AB_CTRL+1, 3$                        ; 0970
                                  00EE   31 0002D 2$:      BRW      10$                                          ;
                                    6B   B5 00030 3$:      TSTW     FDL$AB_FDL_STRING                            ; 0972
                                    F9   13 00032          BEQL     2$                                           ;
                     56             6B   3C 00034          MOVZWL   FDL$AB_FDL_STRING, R6                        ; 0976
                     51        04   AB   D0 00037          MOVL     FDL$AB_FDL_STRING+4, R1                      ; 0977
                     50 00000000G   00   D0 0003B          MOVL     FDL$AB_LINE+4, R0                            ; 0978
```

```
               60          61                  56   28 00042         MOVC3     R6, (R1), (R0)                              ;
              00000000G    00                  56   B0 00046         MOVW      R6, FDL$AB_LINE                             ; 0980
                     01    A9                  20   88 0004D         BISB2     #32, FDL$AB_CTRL+1                          ; 0981
                                               23   11 00051         BRB       5$                                          ; 0962
                              00000000V        00   9F 00053 4$:     PUSHAB    FDL$$READ_ERROR                             ; 0994
                              00000000G        00   9F 00059         PUSHAB    FDL$AB_FDL_RAB                              ;
              00000000G    00                  02   FB 0005F         CALLS     #2, SYS$GET                                 ;
                           59                  50   E9 00066         BLBC      STATUS, 7$                                  ;
              00000000G    00 00000000G        00   B0 00069         MOVW      FDL$AB_FDL_RAB+34, FDL$AB_LINE              ; 0999
                                               DD   13 00074         BEQL      4$                                          ;
              00000000V    00                  00   FB 00076 5$:     CALLS     #0, UPCASE                                  ; 1005
                     04    A8 00000000G        00   D0 0007D         MOVL      FDL$AB_UPCASED+4, FDL$AB_ITEM+4             ; 1009
                           50                  68   3C 00085 6$:     MOVZWL    FDL$AB_ITEM, R0                             ; 1016
                     04    A8                  50   C0 00088         ADDL2     R0, FDL$AB_ITEM+4                           ;
                           51          04      A8   D0 0008C         MOVL      FDL$AB_ITEM+4, R1                           ; 1025
               50 00000000G 00                 51   C3 00090         SUBL3     R1, FDL$AB_UPCASED+4, R0                    ; 1026
                           52 00000000G        00   3C 00098         MOVZWL    FDL$AB_UPCASED, R2                          ; 1024
          FC   AA          50                  52   C1 0009F         ADDL3     R2, R0, FDL$AB_PRE_PARSE_BLOCK+8            ;
                           6A                  51   D0 000A4         MOVL      R1, FDL$AB_PRE_PARSE_BLOCK+12               ; 1028
                     01    A9          C0      8F   8A 000A7         BICB2     #192, FDL$AB_CTRL+1                         ; 1034
                              00000000G        00   9F 000AC         PUSHAB    FDL$AB_PRE_PARSE_KEY                        ; 1037
                              00000000G        00   9F 000B2         PUSHAB    FDL$AB_PRE_PARSE_STATE                      ;
                                       F4      AA   9F 000B8         PUSHAB    FDL$AB_PRE_PARSE_BLOCK                      ;
              00000000G    00                  03   FB 000BB         CALLS     #3, LIB$TPARSE                              ;
                           5B                  50   E9 000C2 7$:     BLBC      STATUS, 11$                                 ;
                           50          04      A8   D0 000C5         MOVL      FDL$AB_ITEM+4, R0                           ; 1041
                     0C    AC                  50   D0 000C9         MOVL      R0, 12(TPARSE_BLOCK)                        ;
          08   AC          6A                  50   C3 000CD         SUBL3     R0, FDL$AB_PRE_PARSE_BLOCK+12, -            ; 1044
                                                                               8(TPARSE_BLOCK)                             ;
                           68          08      AC   B0 000D2         MOVW      8(TPARSE_BLOCK), FDL$AB_ITEM                ; 1046
                           57                  68   3C 000D6         MOVZWL    FDL$AB_ITEM, R7                             ; 1052
                                               03   12 000D9         BNEQ      8$                                          ;
                                             FF45   31 000DB         BRW       1$                                          ;
                              00000000G        00   D6 000DE 8$:     INCL      FDL$GL_STMNTNUM                             ; 1054
               0E          02    A9            02   E1 000E4         BBC       #2, FDL$AB_CTRL+2, 9$                       ; 1058
                           50 00000000G        00   D0 000E9         MOVL      FDL$GL_STNUMPTR, R0                         ; 1060
                           60 00000000G        00   D0 000F0         MOVL      FDL$GL_STMNTNUM, (R0)                       ;
                              00000000G        00   D4 000F7 9$:     CLRL      FDL$GL_SECONDARY                            ; 1064
                              00000000G        00   D4 000FD         CLRL      FDL$GL_SECNUM                               ; 1065
                              00000000G        00   D4 00103         CLRL      FDL$GL_SWITCH                               ; 1066
                              00000000G        00   D4 00109         CLRL      FDL$GL_PROTECTION                           ; 1067
                              00000000G        00   B4 0010F         CLRW      FDL$AB_STRING                               ; 1068
                           69       0308       8F   AA 00115         BICW2     #776, FDL$AB_CTRL                           ; 1071
                           50                  01   D0 0011A         MOVL      #1, R0                                      ; 1073
                                                    04 0011D         RET                                                   ;
                                               50   D4 0011E 10$:    CLRL      R0                                          ; 1075
                                                    04 00120 11$:    RET                                                   ;

; Routine Size:  289 bytes,    Routine Base:  _FDL$CODE + 0000
```

```
1 %SBTTL  'UPCASE'
1 ROUTINE UPCASE : NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Upcases the input line while moving it into the upcase buffer
1 !
1 ! Calling Sequence:
1 !
1 !       UPCASE()
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !
1 !       FDL$AB_LINE     - Descriptor of the input line
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !
1 !       FDL$AB_UPCASED  - Descriptor of the upcased input line
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         CHAR    : REF VECTOR [ ,BYTE ],
2         UPCR    : REF VECTOR [ ,BYTE ];
2
2     ! Point to the string of characters and the upcase buffer
2     !
2     CHAR = .FDL$AB_LINE [ DSC$A_POINTER ];
2     UPCR = .FDL$AB_UPCASED [ DSC$A_POINTER ];
2
2     ! Loop for all the characters in a line
2     !
2     INCR I FROM 0 TO ( .FDL$AB_LINE [ DSC$W_LENGTH ] - 1 ) BY 1
2     DO
2
2         ! If the char. is a lower case letter upcase it
2         ! else just copy it over
2         !
3         IF ( .CHAR [ .I ] GEQU SMALL_A ) AND ( .CHAR [ .I ] LEQU SMALL_Z )
2         THEN
3             UPCR [ .I ] = .CHAR [ .I ] AND ( NOT UPCASE_MASK )
2         ELSE
```

```
;  415        1133  2             UPCR [ .I ] = .CHAR [ .I ];
;  416        1134  2
;  417        1135  2       ! Set the length of the upcased line
;  418        1136  2       !
;  419        1137  2       FDL$AB_UPCASED [ DSC$W_LENGTH ] = .FDL$AB_LINE [ DSC$W_LENGTH ];
;  420        1138  2
;  421        1139  2       RETURN
;  422        1140  2
;  423        1141  1       END;



                                        000C 00000 UPCASE: .WORD    Save R2,R3                                    ; 1077
                     51 00000000G  00     D0 00002         MOVL     FDL$AB_LINE+4, CHAR                           ; 1118
                     50 00000000G  00     D0 00009         MOVL     FDL$AB_UPCASED+4, UPCR                        ; 1119
                     53 00000000G  00     3C 00010         MOVZWL   FDL$AB_LINE, R3                               ; 1123
                     52            01     CE 00017         MNEGL    #1, I                                         ; 1131
                                   1B     11 0001A         BRB      3$                                            ;
                61   8F          6241     91 0001C 1$:     CMPB     (I)[CHAR], #97                                ; 1129
                                   0F     1F 00021         BLSSU    2$                                            ;
                7A   8F          6241     91 00023         CMPB     (I)[CHAR], #122                               ;
                                   08     1A 00028         BGTRU    2$                                            ;
        6240           6241        20     8B 0002A         BICB3    #32, (I)[CHAR], (I)[UPCR]                     ; 1131
                                   05     11 00030         BRB      3$                                            ;
                       6240      6241     90 00032 2$:     MOVB     (I)[CHAR], (I)[UPCR]                          ; 1133
           E1            52        53     F2 00037 3$:     AOBLSS   R3, I, 1$                                     ; 1129
          00000000G      00        53     B0 0003B         MOVW     R3, FDL$AB_UPCASED                            ; 1137
                                   04 00042                RET                                                    ; 1141

; Routine Size:  67 bytes,    Routine Base:  _FDL$CODE + 0121
;  425           1142   1 %SBTTL  'SET_LINE'
;  426           1143   1 GLOBAL ROUTINE  FDL$$SET_LINE =
;  427           1144   1 !++
;  428           1145   1 !
;  429           1146   1 ! Functional Description:
;  430           1147   1 !
;  431           1148   1 ! Calling Sequence:
;  432           1149   1 !
;  433           1150   1 ! Input Parameters:
;  434           1151   1 !       none
;  435           1152   1 !
;  436           1153   1 ! Implicit Inputs:
;  437           1154   1 !       none
;  438           1155   1 !
;  439           1156   1 ! Output Parameters:
;  440           1157   1 !       none
;  441           1158   1 !
;  442           1159   1 ! Implicit Outputs:
;  443           1160   1 !       none
;  444           1161   1 !
;  445           1162   1 ! Routine Value:
;  446           1163   1 !       none
;  447           1164   1 !
;  448           1165   1 ! Side Effects:
;  449           1166   1 !       none
;  450           1167   1 !
;  451           1168   1 !--
;  452           1169   1
;  453           1170   2    BEGIN
;  454           1171   2
;  455           1172   2    TPARSE_ARGS;
;  456           1173   2
;  457           1174   2    FDL$AB_ITEM [ DSC$A_POINTER ] = .TPARSE_BLOCK [ TPA$L_TOKENPTR ];
;  458           1175   2
;  459           1176   2    RETURN SS$_NORMAL
;  460           1177   2
;  461           1178   1    END;



                                            0000 00000         .ENTRY  FDL$$SET_LINE, Save nothing                       ; 1143
                 00000000G  00       14   AC   D0 00002         MOVL    20(TPARSE_BLOCK), FDL$AB_ITEM+4                   ; 1174
                            50            01   D0 0000A         MOVL    #1, R0                                            ; 1176
                                               04 0000D         RET                                                       ; 1178

; Routine Size:  14 bytes,    Routine Base:  _FDL$CODE + 0164
;  463          1179  1 %SBTTL  'SET TERM'
;  464          1180  1 GLOBAL ROUTINE  FDL$$SET_TERM =
;  465          1181  1 !++
;  466          1182  1 !
;  467          1183  1 ! Functional Description:
;  468          1184  1 !
;  469          1185  1 ! Calling Sequence:
;  470          1186  1 !
;  471          1187  1 ! Input Parameters:
;  472          1188  1 !       none
;  473          1189  1 !
;  474          1190  1 ! Implicit Inputs:
;  475          1191  1 !       none
;  476          1192  1 !
;  477          1193  1 ! Output Parameters:
;  478          1194  1 !       none
;  479          1195  1 !
;  480          1196  1 ! Implicit Outputs:
;  481          1197  1 !       none
;  482          1198  1 !
;  483          1199  1 ! Routine Value:
;  484          1200  1 !       none
;  485          1201  1 !
;  486          1202  1 ! Side Effects:
;  487          1203  1 !       none
;  488          1204  1 !
;  489          1205  1 !--
;  490          1206  1
;  491          1207  2    BEGIN
;  492          1208  2
;  493          1209  2    TPARSE_ARGS;
;  494          1210  2
;  495          1211  2    FDL$AB_PRE_PARSE_BLOCK [ TPA$L_STRINGPTR ] =
;  496          1212  2                .FDL$AB_PRE_PARSE_BLOCK [ TPA$L_STRINGPTR ] - 1;
;  497          1213  2
;  498          1214  2    RETURN SS$_NORMAL
;  499          1215  2
;  500          1216  1    END;



                                          0000 00000         .ENTRY  FDL$$SET TERM, Save nothing                    ; 1180
                           00000000G  00   D7 00002         DECL    FDL$AB_PRE_PARSE_BLOCK+12                      ; 1212
                     50               01   D0 00008         MOVL    #1, R0                                         ; 1214
                                           04 0000B         RET                                                    ; 1216

; Routine Size:  12 bytes,    Routine Base:  _FDL$CODE + 0172
```

```
1 %SBTTL  'SET PRIMARY'
1 GLOBAL ROUTINE  FDL$$SET_PRIMARY =
1 !++
1 !
1 ! Functional Description:
1 !
1 ! Calling Sequence:
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     TPARSE_ARGS;
2
2     OWN
2         NXTPRINUM;        ! The next key or area primary number
2
2     LOCAL
2         PRIMASK;
2
2     PRIMASK = .TPARSE_BLOCK [ TPA$L_PARAM ];
2
2     ! If this is the first call then clear an go else check to make sure a
2     ! secondary was processed.
2     !
2     IF .FDL$AB_CTRL [ FDL$V_INITIAL ]
2     THEN
2         FDL$AB_CTRL [ FDL$V_INITIAL ] = _CLEAR
2     ELSE
2
2         ! If a secondary was processed the ok else null primary warning
2         !
2         IF .FDL$AB_CTRL [ FDL$V_SECONDARY ]
2         THEN
2             FDL$AB_CTRL [ FDL$V_SECONDARY ] = _CLEAR
2         ELSE
2             SIGNAL ( FDL$_NULLPRI );
3
3     IF (
4       ( NOT .FDL$AB_CTRL [ FDL$V_DFLT_PRES ] )
3       OR
4       ( .FDL$AB_CTRL [ FDL$V_REPARSE ] )
2       ) THEN
3           BEGIN
3
3           ! If this primary has been defied before check to see if it's a
3           ! key or area primary
3           !
3           IF ( .PRIMASK AND .FDL$AB_PRICTRL ) NEQU 0
3           THEN
3
3               ! Is it a key, area, analysis_of_key or analysis_of_area primary
3               ! check the order in case the last was the same
3               !
4               IF (
5               (
6                   ( .PRIMASK )
5                   AND
6                   ( FDL$M_KEY OR FDL$M_AREA OR FDL$M_ANALK OR FDL$M_ANALA )
4               ) NEQU 0
4
3               ) THEN
3
3                   ! What was the last primary
3                   !
4                   IF (
5                   (.FDL$GL_PRIMARY EQLU FDL$C_KEY)
4                   OR
5                   (.FDL$GL_PRIMARY EQLU FDL$C_AREA)
4                   OR
5                   (.FDL$GL_PRIMARY EQLU FDL$C_ANALK)
4                   OR
5                   (.FDL$GL_PRIMARY EQLU FDL$C_ANALA)
3                   ) THEN
3
3                       ! Check to see if the number is correct
3                       !
3                       IF .FDL$GL_PRINUM EQLU .NXTPRINUM
3                       THEN
3                           NXTPRINUM = .NXTPRINUM + 1
3                       ELSE
4                           BEGIN
4                           SIGNAL( FDL$_OUTORDER,1,.FDL$GL_STMNTNUM );
4                           RETURN FDL$_SYNTAX
4                           END
3
3                   ELSE
3                       NXTPRINUM = 0
3
3               ELSE
3
3                   ! Multiple primaries is only a warning
3                   !
3                   SIGNAL( FDL$_MULPRI,1,.FDL$GL_STMNTNUM )
3
3           ELSE
```

```
                ! Is it a first key or area or ect. primary check the number
                !
                IF ( .PRIMASK AND ( FDL$M_KEY OR FDL$M_AREA OR FDL$M_ANALK OR
                                                        FDL$M_ANALA ) ) NEQU 0
                THEN

                    ! If so check to see if the number is correct
                    !
                    IF .FDL$GL_PRINUM EQLU 0
                    THEN
                        NXTPRINUM = 1
                    ELSE
                        BEGIN
                        SIGNAL( FDL$_OUTORDER,1,.FDL$GL_STMNTNUM );
                        RETURN FDL$_SYNTAX
                        END;

            END;

        ! Flag it for latter
        !
        FDL$AB_PRICTRL = .FDL$AB_PRICTRL OR .PRIMASK;

        ! Clear FDL$PRIMARY so that tparse can set it on return
        !
        FDL$GL_PRIMARY = _CLEAR;

        ! Indicate that a new primary has been found
        !
        FDL$AB_CTRL [ FDL$V_NEWPRI ] = _SET;

        ! Get ready for a new set of secondaries
        !
        INCR I FROM 0 TO (FDL$K_SCTRL_LONG-1)
        DO
            FDL$AB_SECCTRLL [ .I ] = _CLEAR;

        RETURN SS$_NORMAL;

        END;


                                                 .PSECT  _FDL$OWN,NOEXE,  PIC,2

                                00008 NXTPRINUM:
                                                 .BLKB   4


                                                 .PSECT  _FDL$CODE,NOWRT,  SHR,  PIC,2

                          03FC 00000             .ENTRY  FDL$$SET_PRIMARY, Save R2,R3,R4,R5,R6,R7,-  ; 1218
                                                         R8,R9                                       ;
              59 00000000G  00  9E 00002         MOVAB   FDL$GL_STMNTNUM, R9                         ;
              58 00000000G  00  9E 00009         MOVAB   FDL$GL_PRINUM, R8                           ;
```

```
                         57 00000000G  00  9E 00010          MOVAB   FDL$GL_PRIMARY, R7                                   ;
                         56 00000000G  00  9E 00017          MOVAB   FDL$AB_PRICTRL, R6                                   ;
                         55 00000000G  00  9E 0001E          MOVAB   LIB$SIGNAL, R5                                       ;
                         54 00000000'  00  9E 00025          MOVAB   NXTPRINUM, R4                                        ;
                         53 00000000G  00  9E 0002C          MOVAB   FDL$AB_CTRL, R3                                      ;
                         52        20  AC  D0 00033          MOVL    32(TPARSE_BLOCK), PRIMASK                            ; 1255
                                       63  95 00037          TSTB    FDL$AB_CTRL                                          ; 1260
                                       06  18 00039          BGEQ    1$                                                   ;
                         63        80  8F  8A 0003B          BICB2   #128, FDL$AB_CTRL                                    ; 1262
                                       13  11 0003F          BRB     3$                                                   ;
             06          63            06  E1 00041 1$:      BBC     #6, FDL$AB_CTRL, 2$                                  ; 1267
                         63        40  8F  8A 00045          BICB2   #64, FDL$AB_CTRL                                     ; 1269
                                       09  11 00049          BRB     3$                                                   ;
                            00000000G  8F  DD 0004B 2$:      PUSHL   #FDL$_NULLPRI                                        ; 1271
                         65            01  FB 00051          CALLS   #1, LIB$SIGNAL                                       ;
             04        02  A3          01  E1 00054 3$:      BBC     #1, FDL$AB_CTRL+2, 4$                                ; 1274
                         67        02  A3  E9 00059          BLBC    FDL$AB_CTRL+2, 11$                                   ; 1276
                                       50  D4 0005D 4$:      CLRL    R0                                                   ; 1293
                  041C   8F            52  B3 0005F          BITW    PRIMASK, #1052                                       ;
                                       02  13 00064          BEQL    5$                                                   ;
                                       50  D6 00066          INCL    R0                                                   ;
                         66            52  D3 00068 5$:      BITL    PRIMASK, FDL$AB_PRICTRL                              ; 1283
                                       36  13 0006B          BEQL    9$                                                   ;
                         24            50  E9 0006D          BLBC    R0, 8$                                               ; 1289
                         50            67  D0 00070          MOVL    FDL$GL_PRIMARY, R0                                   ; 1301
                         0B            50  D1 00073          CMPL    R0, #11                                              ;
                                       0F  13 00076          BEQL    6$                                                   ;
                         05            50  D1 00078          CMPL    R0, #5                                               ; 1303
                                       0A  13 0007B          BEQL    6$                                                   ;
                         04            50  D1 0007D          CMPL    R0, #4                                               ; 1305
                                       05  13 00080          BEQL    6$                                                   ;
                         03            50  D1 00082          CMPL    R0, #3                                               ; 1307
                                       09  12 00085          BNEQ    7$                                                   ;
                         64            68  D1 00087 6$:      CMPL    FDL$GL_PRINUM, NXTPRINUM                             ; 1312
                                       23  12 0008A          BNEQ    10$                                                  ;
                                       64  D6 0008C          INCL    NXTPRINUM                                            ; 1314
                                       34  11 0008E          BRB     11$                                                  ;
                                       64  D4 00090 7$:      CLRL    NXTPRINUM                                            ; 1322
                                       30  11 00092          BRB     11$                                                  ; 1300
                                       69  DD 00094 8$:      PUSHL   FDL$GL_STMNTNUM                                      ; 1328
                                       01  DD 00096          PUSHL   #1                                                   ;
                            00000000G  8F  DD 00098          PUSHL   #FDL$_MULPRI                                         ;
                         65            03  FB 0009E          CALLS   #3, LIB$SIGNAL                                       ;
                                       21  11 000A1          BRB     11$                                                  ; 1289
                         1E            50  E9 000A3 9$:      BLBC    R0, 11$                                              ; 1335
                                       68  D5 000A6          TSTL    FDL$GL_PRINUM                                        ; 1340
                                       05  12 000A8          BNEQ    10$                                                  ;
                         64            01  D0 000AA          MOVL    #1, NXTPRINUM                                        ; 1342
                                       15  11 000AD          BRB     11$                                                  ;
                                       69  DD 000AF 10$:     PUSHL   FDL$GL_STMNTNUM                                      ; 1345
                                       01  DD 000B1          PUSHL   #1                                                   ;
                            00000000G  8F  DD 000B3          PUSHL   #FDL$_OUTORDER                                       ;
                         65            03  FB 000B9          CALLS   #3, LIB$SIGNAL                                       ;
                         50 00000000G  8F  D0 000BC          MOVL    #FDL$_SYNTAX, R0                                     ; 1346
                                           04 000C3          RET                                                          ;
                         66            52  C8 000C4 11$:     BISL2   PRIMASK, FDL$AB_PRICTRL                              ; 1353
                                       67  D4 000C7          CLRL    FDL$GL_PRIMARY                                       ; 1357
```

```
                                        63            20  88 000C9          BISB2   #32, FDL$AB_CTRL                                  ; 1361
                                                      50  D4 000CC          CLRL    I                                                 ; 1365
                                          00000000G0040  D4 000CE 12$:      CLRL    FDL$AB_SECCTRL[I]                                 ; 1367
                        F5              50            05  F3 000D5          AOBLEQ  #5, I, 12$                                        ;
                                        50            01  D0 000D9          MOVL    #1, R0                                            ; 1369
                                                          04 000DC          RET                                                       ; 1371

; Routine Size:  221 bytes,    Routine Base:  _FDL$CODE + 017E
```

```
1 %SBTTL  'SET_SECONDARY'
1 GLOBAL ROUTINE  FDL$$SET_SECONDARY =
1 !++
1 !
1 ! Functional Description:
1 !
1 ! Calling Sequence:
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     TPARSE_ARGS;
2
2     LOCAL
2         SECBIT  : LONG;
2
2     SECBIT = .TPARSE_BLOCK [ TPA$L_PARAM ];
2
2     ! See if the secondary has been defined before
2     !
2     IF .FDL$AB_SECCTRL [ .SECBIT ]
2     THEN
2
2         ! If it has then see if it was a key segment thing
2         !
3         IF (
4         ( .SECBIT EQLU FDL$C_SEGPOS )
3         OR
4         ( .SECBIT EQLU FDL$C_SEGLEN )
3         OR
4         ( .SECBIT EQLU FDL$C_SEGTYP )
2         ) THEN
3             BEGIN
3
3             ! If it's out of bounds it's an error
3             !
3             IF .FDL$GL_SECNUM GTR 7
3             THEN
```

```
;  715       1429  4                     BEGIN
;  716       1430  4                     SIGNAL( FDL$_UNSECKW,3,
;  717       1431  4                             .FDL$GL_STMNTNUM,
;  718       1432  4                             .TPARSE_BLOCK [ TPA$L_TOKENCNT ],
;  719       1433  4                             .TPARSE_BLOCK [ TPA$L_TOKENPTR ] );
;  720       1434  4                     RETURN FDL$_SYNTAX
;  721       1435  4                     END
;  722       1436  3                 END
;  723       1437  2             ELSE
;  724       1438  2
;  725       1439  2                 ! If it has been defined before it's only a warning
;  726       1440  2                 !
;  727       1441  2                 SIGNAL( FDL$_MULSEC,1,.FDL$GL_STMNTNUM )
;  728       1442  2         ELSE
;  729       1443  2
;  730       1444  2             ! Flag it for next time (unless it's an ACL ENTRY - which can be repeated)
;  731       1445  2             !
;  732       1446  2             IF .SECBIT NEQU FDL$C_ACE
;  733       1447  2             THEN
;  734       1448  2                 FDL$AB_SECCTRL [ .SECBIT ] = _SET;
;  735       1449  2
;  736       1450  2         ! Get ready for a new an wonderous qualifier
;  737       1451  2         !
;  738       1452  2         FDL$GL_QUALIFIER = _CLEAR;
;  739       1453  2
;  740       1454  2         RETURN SS$_NORMAL
;  741       1455  2
;  742       1456  1         END;
```

```
                                          001C 00000          .ENTRY  FDL$$SET_SECONDARY, Save R2,R3,R4                ; 1373
                   54 00000000G     00    9E 00002          MOVAB   FDL$AB_SECCTRL, R4                                 ;
                   53 00000000G     00    9E 00009          MOVAB   LIB$SIGNAL, R3                                     ;
                   52 00000000G     00    9E 00010          MOVAB   FDL$GL_STMNTNUM, R2                                ;
                   50          20   AC    D0 00017          MOVL    32(TPARSE_BLOCK), SECBIT                           ; 1407
          4C       64               50    E1 0001B          BBC     SECBIT, FDL$AB_SECCTRL, 3$                         ; 1411
             00000086 8F            50    D1 0001F          CMPL    SECBIT, #134                                       ; 1417
                                    12    13 00026          BEQL    1$                                                 ;
             00000085 8F            50    D1 00028          CMPL    SECBIT, #133                                       ; 1419
                                    09    13 0002F          BEQL    1$                                                 ;
             00000087 8F            50    D1 00031          CMPL    SECBIT, #135                                       ; 1421
                                    22    12 00038          BNEQ    2$                                                 ;
                   07 00000000G     00    D1 0003A 1$:      CMPL    FDL$GL_SECNUM, #7                                  ; 1427
                                    31    15 00041          BLEQ    4$                                                 ;
                   7E          10   AC    7D 00043          MOVQ    16(TPARSE_BLOCK), -(SP)                            ; 1432
                                    62    DD 00047          PUSHL   FDL$GL_STMNTNUM                                    ; 1431
                                    03    DD 00049          PUSHL   #3                                                 ; 1430
                      00000000G     8F    DD 0004B          PUSHL   #FDL$_UNSECKW                                      ;
                   63               05    FB 00051          CALLS   #5, LIB$SIGNAL                                     ;
                   50 00000000G     8F    D0 00054          MOVL    #FDL$_SYNTAX, R0                                   ; 1434
                                          04 0005B          RET                                                        ;
                                    62    DD 0005C 2$:      PUSHL   FDL$GL_STMNTNUM                                    ; 1441
                                    01    DD 0005E          PUSHL   #1                                                 ;
                      00000000G     8F    DD 00060          PUSHL   #FDL$_MULSEC                                       ;
```

```
                        63            03  FB 00066        CALLS   #3, LIB$SIGNAL                          ;
                                      09  11 00069        BRB     4$                                      ; 1416
                        08            50  D1 0006B 3$:    CMPL    SECBIT, #8                              ; 1446
                                      04  13 0006E        BEQL    4$                                      ;
           00           64            50  E2 00070        BBSS    SECBIT, FDL$AB_SECCTRL, 4$              ; 1448
                           00000000G  00  D4 00074 4$:    CLRL    FDL$GL_QUALIFIER                        ; 1452
                        50            01  D0 0007A        MOVL    #1, R0                                  ; 1454
                                          04 0007D        RET                                             ; 1456

; Routine Size:  126 bytes,    Routine Base:  _FDL$CODE + 025B
;  744           1457  1 %SBTTL  'START_STR'
;  745           1458  1 GLOBAL ROUTINE  FDL$$START_STR =
;  746           1459  1 !++
;  747           1460  1 !
;  748           1461  1 ! Functional Description:
;  749           1462  1 !
;  750           1463  1 !       Initializes the string descriptor
;  751           1464  1 !
;  752           1465  1 ! Calling Sequence:
;  753           1466  1 !
;  754           1467  1 !       Called from the parse tables
;  755           1468  1 !
;  756           1469  1 ! Input Parameters:
;  757           1470  1 !       none
;  758           1471  1 !
;  759           1472  1 ! Implicit Inputs:
;  760           1473  1 !       none
;  761           1474  1 !
;  762           1475  1 ! Output Parameters:
;  763           1476  1 !       none
;  764           1477  1 !
;  765           1478  1 ! Implicit Outputs:
;  766           1479  1 !       none
;  767           1480  1 !
;  768           1481  1 ! Routine Value:
;  769           1482  1 !       none
;  770           1483  1 !
;  771           1484  1 ! Side Effects:
;  772           1485  1 !       none
;  773           1486  1 !
;  774           1487  1 !--
;  775           1488  1
;  776           1489  2    BEGIN
;  777           1490  2
;  778           1491  2    TPARSE_ARGS;
;  779           1492  2
;  780           1493  2    ! Start the makings of a descriptor
;  781           1494  2    !
;  782           1495  2    FDL$AB_STRING [ DSC$A_POINTER ] = .TPARSE_BLOCK [ TPA$L_TOKENPTR ];
;  783           1496  2
;  784           1497  2    ! Process blanks
;  785           1498  2    !
;  786           1499  2    TPARSE_BLOCK [ TPA$V_BLANKS ] = _SET;
;  787           1500  2
;  788           1501  2    RETURN SS$_NORMAL
;  789           1502  2
;  790           1503  1    END;



                                           0000 00000        .ENTRY  FDL$$START_STR, Save nothing                  ; 1458
          00000000G  00        14     AC   D0 00002          MOVL    20(TPARSE_BLOCK), FDL$AB_STRING+4             ; 1495
                 04  AC               01   88 0000A          BISB2   #1, 4(TPARSE_BLOCK)                           ; 1499
                     50               01   D0 0000E          MOVL    #1, R0                                        ; 1501
                                           04 00011          RET                                                   ; 1503
```

```
; Routine Size:  18 bytes,    Routine Base:  _FDL$CODE + 02D9
```

```
1 %SBTTL  'END_STR'
1 GLOBAL ROUTINE  FDL$$END_STR =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Terminates the processing of a string and determines the length
1 !
1 ! Calling Sequence:
1 !
1 !       Called from the parse tables
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         SAVE_LEN        : WORD,
2         CUT_LEN         : WORD;
2
2     TPARSE_ARGS;
2
2     TPARSE_BLOCK [ TPA$V_BLANKS ] = _CLEAR;
2
2     ! The size is from where we are minus from where we is
2     !
2     FDL$AB_STRING [ DSC$W_LENGTH ] = .TPARSE_BLOCK [ TPA$L_STRINGPTR ] -
2                                         .FDL$AB_STRING [ DSC$A_POINTER ];
2
2     ! If the last char was a "!" then subtract one
2     !
2     IF .TPARSE_BLOCK [ TPA$B_CHAR ] EQL COMMENT_MARK
2     THEN
2         FDL$AB_STRING [ DSC$W_LENGTH ] = .FDL$AB_STRING [ DSC$W_LENGTH ] - 1;
2
2     ! Save this length
2     !
2     SAVE_LEN = .FDL$AB_STRING [ DSC$W_LENGTH ];
2
```

```
;   849          1561  2       ! Remove trailing blanks
;   850          1562  2       !
;   851          1563  2       STR$TRIM ( FDL$AB_STRING,FDL$AB_STRING,CUT_LEN );
;   852          1564  2
;   853          1565  2       ! Set the trimmed length
;   854          1566  2       !
;   855          1567  2       FDL$AB_STRING [ DSC$W_LENGTH ] = .CUT_LEN;
;   856          1568  2
;   857          1569  2       ! Remove any leading white space from the string
;   858          1570  2       !
;   859          1571  2       FDL$AB_STRING [ DSC$W_LENGTH ] = TRIM_LEADING ();
;   860          1572  2
;   861          1573  2       ! Remove any quotes from the upcased string
;   862          1574  2       !
;   863          1575  2       FDL$AB_STRING [ DSC$W_LENGTH ] = EXTRACT_QUOTE ();
;   864          1576  2
;   865          1577  2       ! Adjust the pointer so that we are looking into the original input line
;   866          1578  2       !
;   867          1579  2       FDL$AB_STRING [ DSC$A_POINTER ] = .FDL$AB_STRING [ DSC$A_POINTER ] -
;   868          1580  2                                                     .FDL$GL_MAXLINE;
;   869          1581  2
;   870          1582  2       ! Restore the original length
;   871          1583  2       !
;   872          1584  2       FDL$AB_STRING [ DSC$W_LENGTH ] = .SAVE_LEN;
;   873          1585  2
;   874          1586  2       ! Remove trailing blanks
;   875          1587  2       !
;   876          1588  2       STR$TRIM ( FDL$AB_STRING,FDL$AB_STRING,CUT_LEN );
;   877          1589  2
;   878          1590  2       ! Set the trimmed length
;   879          1591  2       !
;   880          1592  2       FDL$AB_STRING [ DSC$W_LENGTH ] = .CUT_LEN;
;   881          1593  2
;   882          1594  2       ! Remove any leading white space from the string
;   883          1595  2       !
;   884          1596  2       FDL$AB_STRING [ DSC$W_LENGTH ] = TRIM_LEADING ();
;   885          1597  2
;   886          1598  2       ! Remove any quotes from the original string
;   887          1599  2       !
;   888          1600  2       FDL$AB_STRING [ DSC$W_LENGTH ] = EXTRACT_QUOTE ();
;   889          1601  2
;   890          1602  2       RETURN SS$_NORMAL;
;   891          1603  2
;   892          1604  1       END;



                                    007C 00000         .ENTRY  FDL$$END_STR, Save R2,R3,R4,R5,R6             ; 1505
                56 00000000V    00    9E 00002         MOVAB   EXTRACT_QUOTE, R6                                  ;
                55 00000000V    00    9E 00009         MOVAB   TRIM_LEADING, R5                                   ;
                54 00000000G    00    9E 00010         MOVAB   STR$TRIM, R4                                       ;
                53 00000000G    00    9E 00017         MOVAB   FDL$AB_STRING, R3                                  ;
                5E              04    C2 0001E         SUBL2   #4, SP                                             ;
          04    AC              01    8A 00021         BICB2   #1, 4(TPARSE_BLOCK)                                ; 1544
   63     0C    AC      04      A3    A3 00025         SUBW3   FDL$AB_STRING+4, 12(TPARSE_BLOCK), -               ; 1549
```

```
                                                                                FDL$AB_STRING
                         21          18    AC   91 0002B           CMPB        24(TPARSE_BLOCK), #33                            ; 1553
                                           02   12 0002F           BNEQ        1$                                               ;
                                           63   B7 00031           DECW        FDL$AB_STRING                                    ; 1555
                         52                63   B0 00033 1$:       MOVW        FDL$AB_STRING, SAVE_LEN                          ; 1559
                                   4008    8F   BB 00036           PUSHR       #^M<R3,SP>                                       ; 1563
                                           53   DD 0003A           PUSHL       R3                                               ;
                         64                03   FB 0003C           CALLS       #3, STR$TRIM                                     ;
                         63                6E   B0 0003F           MOVW        CUT_LEN, FDL$AB_STRING                           ; 1567
                         65                00   FB 00042           CALLS       #0, TRIM_LEADING                                 ; 1571
                         63                50   B0 00045           MOVW        R0, FDL$AB_STRING                                ;
                         66                00   FB 00048           CALLS       #0, EXTRACT_QUOTE                                ; 1575
                         63                50   B0 0004B           MOVW        R0, FDL$AB_STRING                                ;
                  04     A3   00000000G    00   C2 0004E           SUBL2       FDL$GL_MAXLINE, FDL$AB_STRING+4                  ; 1580
                         63                52   B0 00056           MOVW        SAVE_LEN, FDL$AB_STRING                          ; 1584
                                   4008    8F   BB 00059           PUSHR       #^M<R3,SP>                                       ; 1588
                                           53   DD 0005D           PUSHL       R3                                               ;
                         64                03   FB 0005F           CALLS       #3, STR$TRIM                                     ;
                         63                6E   B0 00062           MOVW        CUT_LEN, FDL$AB_STRING                           ; 1592
                         65                00   FB 00065           CALLS       #0, TRIM_LEADING                                 ; 1596
                         63                50   B0 00068           MOVW        R0, FDL$AB_STRING                                ;
                         66                00   FB 0006B           CALLS       #0, EXTRACT_QUOTE                                ; 1600
                         63                50   B0 0006E           MOVW        R0, FDL$AB_STRING                                ;
                         50                01   D0 00071           MOVL        #1, R0                                           ; 1602
                                                04 00074           RET                                                          ; 1604

; Routine Size:  117 bytes,    Routine Base:  _FDL$CODE + 02EB
```

```
1 %SBTTL  'EXTRACT_QUOTE'
1 ROUTINE EXTRACT_QUOTE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       It also extracts out embedded or bracketing quotes or apostrophes
1 !
1 ! Calling Sequence:
1 !
1 !       Called from END_STR
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       The new string length - after the quotes are removed.
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         QCHAR   : BYTE,
2         J       : LONG,
2         NEW_LEN : LONG,
2         CUT_LEN : LONG,
2         STR     : REF VECTOR [ ,BYTE ],
2         TMP_STR : REF VECTOR [ ,BYTE ];
2
2     NEW_LEN = .FDL$AB_STRING [ DSC$W_LENGTH ];
2
2     ! Now extract out any bracketing or embedded quotes or apostrophes
2     !
2     IF .FDL$AB_CTRL [ FDL$V_QUOTE_PRES ] OR .FDL$AB_CTRL [ FDL$V_APOST_PRES ]
2     THEN
3         BEGIN
3
3         CUT_LEN = .FDL$AB_STRING [ DSC$W_LENGTH ];
3         TMP_STR = FDL$$GET_VM ( .CUT_LEN );
3
3         STR = .FDL$AB_STRING [ DSC$A_POINTER ];
3
3         IF .FDL$AB_CTRL [ FDL$V_QUOTE_PRES ]
3         THEN
```

```
3                       QCHAR = '"'
3                   ELSE IF .FDL$AB_CTRL [ FDL$V_APOST_PRES ]
3                   THEN
3                       QCHAR = '''';
3
3                   CH$MOVE ( .CUT_LEN,.FDL$AB_STRING [ DSC$A_POINTER ],.TMP_STR );
3
3                   NEW_LEN = 0;
3                   J = 0;
3
4                   WHILE .J LEQ (.CUT_LEN - 1)
3                   DO
4                       BEGIN
4
4                       ! Now copy the string back, but stripping the QCHARs
4                       ! according to the rules that embedded "" ==> " and '' ==> '
4                       !
4                       IF .TMP_STR [ .J ] EQLU .QCHAR
4                       THEN
5                           BEGIN
5
5                           ! If we're not at the beginning or end of the string,
5                           ! copy one qchar and skip the next
5                           !
6                           IF NOT ((.J EQLU 0) OR (.J EQLU (.CUT_LEN-1)))
5                           THEN
6                               BEGIN
6
6                               IF .TMP_STR [ .J+1 ] EQLU .QCHAR
6                               THEN
6                                   J = .J + 1;
6
6                               STR [ .NEW_LEN ] = .TMP_STR [ .J ];
6                               NEW_LEN = .NEW_LEN + 1
6
5                               END;
5                           END
5
4                       ELSE
4                           ! Just copy the character back and bump the count
4                           !
5                           BEGIN
5                           STR [ .NEW_LEN ] = .TMP_STR [ .J ];
5                           NEW_LEN = .NEW_LEN + 1
4                           END;
4
4                       J = .J + 1;
4
3                       END;            ! do
3
3                   ! Release the tmp string
3                   !
3                   FDL$$FREE_VM ( .CUT_LEN,.TMP_STR );
3
2                   END;     ! IF QUOTE OR APOST PRESENT
2
2               ! The routine value is the new length
```

```
; 1008           1719  2       !
; 1009           1720  2       RETURN .NEW_LEN;
; 1010           1721  2
; 1011           1722  1       END;




                                                       0FFC 00000 EXTRACT_QUOTE:
                                                                          .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11                      ; 1606
                            5B 00000000G  00    9E 00002                  MOVAB   FDL$AB_CTRL, R11                                           ;
                            50 00000000G  00    3C 00009                  MOVZWL  FDL$AB_STRING, R0                                          ; 1647
                            5A               50 D0 00010                  MOVL    R0, NEW_LEN                                                ;
                                     01      AB 95 00013                  TSTB    FDL$AB_CTRL+1                                              ; 1651
                                             05 19 00016                  BLSS    1$                                                         ;
             67         01  AB               06 E1 00018                  BBC     #6, FDL$AB_CTRL+1, 8$                                      ;
                            56               50 D0 0001D 1$:              MOVL    R0, CUT_LEN                                                ; 1655
                                             56 DD 00020                  PUSHL   CUT_LEN                                                    ; 1656
               00000000G    00               01 FB 00022                  CALLS   #1, FDL$$GET_VM                                            ;
                            57               50 D0 00029                  MOVL    R0, TMP_STR                                                ;
                            50 00000000G  00    D0 0002C                  MOVL    FDL$AB_STRING+4, R0                                        ; 1658
                            58               50 D0 00033                  MOVL    R0, STR                                                    ;
                                     01      AB 95 00036                  TSTB    FDL$AB_CTRL+1                                              ; 1660
                                             05 18 00039                  BGEQ    2$                                                         ;
                            59               22 90 0003B                  MOVB    #34, QCHAR                                                 ; 1662
                                             08 11 0003E                  BRB     3$                                                         ;
             03         01  AB               06 E1 00040 2$:              BBC     #6, FDL$AB_CTRL+1, 3$                                      ; 1663
                            59               27 90 00045                  MOVB    #39, QCHAR                                                 ; 1665
             67             60               56 28 00048 3$:              MOVC3   CUT_LEN, (R0), (TMP_STR)                                   ; 1667
                                             5A D4 0004C                  CLRL    NEW_LEN                                                    ; 1669
                                             50 D4 0004E                  CLRL    J                                                          ; 1670
                            51       FF      A6 9E 00050                  MOVAB   -1(R6), R1                                                 ; 1672
                            51               50 D1 00054 4$:              CMPL    J, R1                                                      ;
                                             21 14 00057                  BGTR    7$                                                         ;
                            59             6047 91 00059                  CMPB    (J)[TMP_STR], QCHAR                                        ; 1679
                                             12 12 0005D                  BNEQ    5$                                                         ;
                                             50 D5 0005F                  TSTL    J                                                          ; 1686
                                             13 13 00061                  BEQL    6$                                                         ;
                            51               50 D1 00063                  CMPL    J, R1                                                      ;
                                             0E 13 00066                  BEQL    6$                                                         ;
                            59          01 A047 91 00068                  CMPB    1(J)[TMP_STR], QCHAR                                       ; 1690
                                             02 12 0006D                  BNEQ    5$                                                         ;
                                             50 D6 0006F                  INCL    J                                                          ; 1692
                          BA48             6047 90 00071 5$:              MOVB    (J)[TMP_STR], (NEW_LEN)+[STR]                              ; 1704
                                             50 D6 00076 6$:              INCL    J                                                          ; 1708
                                             DA 11 00078                  BRB     4$                                                         ; 1672
                            7E               56 7D 0007A 7$:              MOVQ    CUT_LEN, -(SP)                                             ; 1714
               00000000G    00               02 FB 0007D                  CALLS   #2, FDL$$FREE_VM                                           ;
                            50               5A D0 00084 8$:              MOVL    NEW_LEN, R0                                                ; 1720
                                                04 00087                  RET                                                                ; 1722

; Routine Size:  136 bytes,    Routine Base:  _FDL$CODE + 0360
```

```
%SBTTL  'TRIM_LEADING'
ROUTINE TRIM_LEADING =
!++
!
! Functional Description:
!
!       It removes leading spaces and tabs from the input string
!
! Calling Sequence:
!
!       Called from END_STR
!
! Input Parameters:
!       none
!
! Implicit Inputs:
!       none
!
! Output Parameters:
!       none
!
! Implicit Outputs:
!       none
!
! Routine Value:
!       The new string length - after the white space is removed.
!
! Side Effects:
!       none
!
!--

    BEGIN

    LOCAL
        FLAG    : BYTE,
        TMP     : BYTE,
        BLANK   : BYTE,
        TAB     : BYTE,
        J       : LONG,
        NEW_LEN : LONG,
        CUT_LEN : LONG,
        STR     : REF VECTOR [ ,BYTE ],
        TMP_STR : REF VECTOR [ ,BYTE ];

    BLANK = ' ';
    TAB = '	';
    TMP = ..FDL$AB_STRING [ DSC$A_POINTER ];

    NEW_LEN = .FDL$AB_STRING [ DSC$W_LENGTH ];

    ! Now extract out any bracketing or embedded quotes or apostrophes
    !
    IF (.TMP EQLU .BLANK) OR (.TMP EQLU .TAB)
    THEN
        BEGIN
```

```
3          CUT_LEN = .FDL$AB_STRING [ DSC$W_LENGTH ];
3          TMP_STR = FDL$$GET_VM ( .CUT_LEN );
3
3          STR = .FDL$AB_STRING [ DSC$A_POINTER ];
3
3          CH$MOVE ( .CUT_LEN,.FDL$AB_STRING [ DSC$A_POINTER ],.TMP_STR );
3
3          NEW_LEN = 0;
3          J = 0;
3          FLAG = _CLEAR;
3
4          WHILE .J LEQ (.CUT_LEN - 1)
3          DO
4              BEGIN
4
4              ! Now copy the string back, but stripping the white space
4              !
5              IF (.TMP_STR [ .J ] EQLU .BLANK) OR (.TMP_STR [ .J ] EQLU .TAB)
4              THEN
5                  BEGIN
5
5                  ! If we have seen the a non-white character
5                  ! just copy this blank or tab like any other char
5                  !
5                  IF .FLAG
5                  THEN
6                      BEGIN
6
6                      STR [ .NEW_LEN ] = .TMP_STR [ .J ];
6                      NEW_LEN = .NEW_LEN + 1
6
5                      END;
5                  END
4              ELSE
4                  ! Just copy the character back and bump the count
4                  !
5                  BEGIN
5
5                  FLAG = _SET;
5                  STR [ .NEW_LEN ] = .TMP_STR [ .J ];
5                  NEW_LEN = .NEW_LEN + 1
5
4                  END;
4
4              J = .J + 1;
4
3              END;          ! do
3
3          ! Release the tmp string
3          !
3          FDL$$FREE_VM ( .CUT_LEN,.TMP_STR );
3
2          END;     ! IF THERE IS LEADING WHITE SPACE
2
2      ! The routine value is the new length
2      !
2      RETURN .NEW_LEN;
```

```
                                                          E 11
FDLDRIVER          VAX-11 FDL Utilities                  16-Sep-1984 01:47:45    VAX-11 Bliss-32 V4.0-742              Page  33
V04-000            TRIM_LEADING                          14-Sep-1984 12:31:17    [FDL.SRC]FDLDRIVER.B32;1                    (13)

; 1127             1837  2
; 1128             1838  1       END;




                                                   0FFC 00000 TRIM_LEADING:
                                                                   .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11           ; 1724
                             5B            20   90 00002           MOVB     #32, BLANK                                     ; 1768
                             5A            09   90 00005           MOVB     #9, TAB                                        ; 1769
                             50 00000000G  00   D0 00008           MOVL     FDL$AB_STRING+4, R0                            ; 1770
                             51            60   90 0000F           MOVB     (R0), TMP                                      ;
                             50 00000000G  00   3C 00012           MOVZWL   FDL$AB_STRING, R0                              ; 1772
                             59            50   D0 00019           MOVL     R0, NEW_LEN                                    ;
                             5B            51   91 0001C           CMPB     TMP, BLANK                                     ; 1776
                                           05   13 0001F           BEQL     1$                                             ;
                             5A            51   91 00021           CMPB     TMP, TAB                                       ;
                                           53   12 00024           BNEQ     8$                                             ;
                             56            50   D0 00026 1$:       MOVL     R0, CUT_LEN                                    ; 1780
                                           56   DD 00029           PUSHL    CUT_LEN                                        ; 1781
                   00000000G  00           01   FB 0002B           CALLS    #1, FDL$$GET_VM                                ;
                             57            50   D0 00032           MOVL     R0, TMP_STR                                    ;
                             50 00000000G  00   D0 00035           MOVL     FDL$AB_STRING+4, R0                            ; 1783
                             58            50   D0 0003C           MOVL     R0, STR                                        ;
                67           60            56   28 0003F           MOVC3    CUT_LEN, (R0), (TMP_STR)                       ; 1785
                                           59   D4 00043           CLRL     NEW_LEN                                        ; 1787
                                           50   D4 00045           CLRL     J                                              ; 1788
                                           52   94 00047           CLRB     FLAG                                           ; 1789
                             51        FF  A6   9E 00049           MOVAB    -1(R6), R1                                     ; 1791
                             51            50   D1 0004D 2$:       CMPL     J, R1                                          ;
                                           1D   14 00050           BGTR     7$                                             ;
                             5B          6047   91 00052           CMPB     (J)[TMP_STR], BLANK                            ; 1797
                                           06   13 00056           BEQL     3$                                             ;
                             5A          6047   91 00058           CMPB     (J)[TMP_STR], TAB                              ;
                                           05   12 0005C           BNEQ     4$                                             ;
                             0A            52   E9 0005E 3$:       BLBC     FLAG, 6$                                       ; 1804
                                           03   11 00061           BRB      5$                                             ; 1808
                             52            01   90 00063 4$:       MOVB     #1, FLAG                                       ; 1818
                           8948          6047   90 00066 5$:       MOVB     (J)[TMP_STR], (NEW_LEN)+[STR]                  ; 1819
                                           50   D6 0006B 6$:       INCL     J                                              ; 1824
                                           DE   11 0006D           BRB      2$                                             ; 1791
                             7E            56   7D 0006F 7$:       MOVQ     CUT_LEN, -(SP)                                 ; 1830
                   00000000G  00           02   FB 00072           CALLS    #2, FDL$$FREE_VM                               ;
                             50            59   D0 00079 8$:       MOVL     NEW_LEN, R0                                    ; 1836
                                                04 0007C           RET                                                     ; 1838

; Routine Size:  125 bytes,    Routine Base:  _FDL$CODE + 03E8
```

```
1 %SBTTL  'SET DATE TIME'
1 GLOBAL ROUTINE  FDL$$SET_DATE_TIME =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Sets up the date/time quadword
1 !
1 ! Calling Sequence:
1 !
1 !       Called from the parse tables
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2    TPARSE_ARGS;
2
2    LOCAL
2        TEMP_DESC        : DESC_BLK;
2
2    ! We must adjust the pointer so it points to the upcased buffer
2    !
2    TEMP_DESC [ DSC$W_LENGTH ] = .FDL$AB_STRING [ DSC$W_LENGTH ];
2    TEMP_DESC [ DSC$A_POINTER ] = .FDL$AB_STRING [ DSC$A_POINTER ] +
2                                        .FDL$GL_MAXLINE;
2
2    ! If there is an error signal it and return failure
2    !
2    IF NOT SYS$BINTIM( TEMP_DESC,FDL$AL_DATE_TIME )
2    THEN
3        BEGIN
3
3        BUILTIN CALLG;
3
3        TPARSE_BLOCK [ TPA$L_PARAM ] = FDL$_INVDATIM;
3
3        CALLG( .TPARSE_BLOCK,FDL$$SYNTAX_ERROR );
3
```

```
; 1187          1896  3              RETURN 0
; 1188          1897  3
; 1189          1898  2              END;
; 1190          1899  2
; 1191          1900  2          RETURN SS$_NORMAL
; 1192          1901  2
; 1193          1902  1          END;



                                                       0000 00000          .ENTRY  FDL$$SET_DATE_TIME, Save nothing                ; 1840
                              5E                  08  C2 00002            SUBL2   #8, SP                                           ;
                              6E 00000000G        00  B0 00005            MOVW    FDL$AB_STRING, TEMP_DESC                         ; 1880
          04   AE 00000000G  00 00000000G        00  C1 0000C            ADDL3   FDL$GL_MAXLINE, FDL$AB_STRING+4, -               ; 1882
                                                                                  TEMP_DESC+4
                                 00000000G        00  9F 00019            PUSHAB  FDL$AL_DATE_TIME                                 ; 1886
                                        04        AE  9F 0001F            PUSHAB  TEMP_DESC                                        ;
                 00000000G    00                  02  FB 00022            CALLS   #2, SYS$BINTIM                                   ;
                              11                  50  E8 00029            BLBS    R0, 1$                                           ;
                 20           AC 00000000G        8F  D0 0002C            MOVL    #FDL$_INVDATIM, 32(TPARSE_BLOCK)                 ; 1892
                 00000000V    00                  6C  FA 00034            CALLG   (TPARSE_BLOCK), FDL$$SYNTAX_ERROR                ; 1894
                                                  04  11 0003B            BRB     2$                                               ; 1896
                              50                  01  D0 0003D  1$:       MOVL    #1, R0                                           ; 1900
                                                      04 00040            RET                                                      ;
                                                  50  D4 00041  2$:       CLRL    R0                                               ; 1902
                                                      04 00043            RET                                                      ;

; Routine Size:  68 bytes,    Routine Base:  _FDL$CODE + 0465
; 1195          1903  1 %SBTTL  'SET COMMENT'
; 1196          1904  1 GLOBAL ROUTINE  FDL$$SET_COMMENT =
; 1197          1905  1 !++
; 1198          1906  1 !
; 1199          1907  1 ! Functional Description:
; 1200          1908  1 !
; 1201          1909  1 !       Sets up the comment descriptor
; 1202          1910  1 !
; 1203          1911  1 ! Calling Sequence:
; 1204          1912  1 !
; 1205          1913  1 !       Called from the parse tables
; 1206          1914  1 !
; 1207          1915  1 ! Input Parameters:
; 1208          1916  1 !       none
; 1209          1917  1 !
; 1210          1918  1 ! Implicit Inputs:
; 1211          1919  1 !       none
; 1212          1920  1 !
; 1213          1921  1 ! Output Parameters:
; 1214          1922  1 !       none
; 1215          1923  1 !
; 1216          1924  1 ! Implicit Outputs:
; 1217          1925  1 !       none
; 1218          1926  1 !
; 1219          1927  1 ! Routine Value:
; 1220          1928  1 !       none
; 1221          1929  1 !
; 1222          1930  1 ! Side Effects:
; 1223          1931  1 !       none
; 1224          1932  1 !
; 1225          1933  1 !--
; 1226          1934  1
; 1227          1935  2     BEGIN
; 1228          1936  2
; 1229          1937  2     TPARSE_ARGS;
; 1230          1938  2
; 1231          1939  2     ! The comment is the rest of the line
; 1232          1940  2     !
; 1233          1941  2     FDL$AB_COMMENT [ DSC$W_LENGTH ] = .TPARSE_BLOCK [ TPA$L_STRINGCNT ] + 1;
; 1234          1942  2     FDL$AB_COMMENT [ DSC$A_POINTER ] = .TPARSE_BLOCK [ TPA$L_STRINGPTR ] - 1;
; 1235          1943  2
; 1236          1944  2     ! Adjust the pointer so that we are looking into the original input line
; 1237          1945  2     !
; 1238          1946  2     FDL$AB_COMMENT [ DSC$A_POINTER ] = .FDL$AB_COMMENT [ DSC$A_POINTER ] -
; 1239          1947  2                                                     .FDL$GL_MAXLINE;
; 1240          1948  2
; 1241          1949  2     RETURN SS$_NORMAL
; 1242          1950  2
; 1243          1951  1     END;



                                         0004 00000          .ENTRY  FDL$$SET_COMMENT, Save R2                      ; 1904
                            52 00000000G  00   9E 00002          MOVAB   FDL$AB_COMMENT+4, R2                           ;
         FC    A2       08  AC            01   A1 00009          ADDW3   #1, 8(TPARSE_BLOCK), FDL$AB_COMMENT            ; 1941
```

```
                    62        0C    AC                01  C3 0000F        SUBL3   #1, 12(TPARSE_BLOCK), FDL$AB_COMMENT+4        ; 1942
                                    62  00000000G     00  C2 00014        SUBL2   FDL$GL_MAXLINE, FDL$AB_COMMENT+4              ; 1947
                                    50                01  D0 0001B        MOVL    #1, R0                                        ; 1949
                                                          04 0001E        RET                                                   ; 1951

; Routine Size:  31 bytes,    Routine Base:  _FDL$CODE + 04A9
```

```
1 %SBTTL  'SYNTAX_ERRROR'
1 GLOBAL ROUTINE  FDL$$SYNTAX_ERROR =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Syntax_error has two functions: If called with the argument fdl$_abkw
1 !       or fdl$_abprikw it checks if there has been an ambigous keyword, if
1 !       there has been then it signals the error else it returns failure. If
1 !       it is called with some other error it is signaled and return is normal.
1 !
1 ! Calling Sequence:
1 !
1 !       Called from the parse tables
1 !
1 !       Can be called from a bliss routine by:
1 !
1 !       BUILTIN CALLG;
1 !
1 !       CALLG( tparse_block,FDL$$SYNTAX_ERROR )
1 !
1 ! Input Parameters:
1 !
1 !       Error code in the tpa$l_param field of the tparse_block
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !
1 !       ss$_normal or 0 (see above)
1 !
1 ! Side Effects:
1 !
1 !       Signals an error
1 !
1 !--
1
2    BEGIN
2
2    TPARSE_ARGS;
2
2    LOCAL       STATUS  : LONG;
2
2    BIND        CODE = STATUS : BLOCK [ 4,BYTE ];
2
2    ! Get the error code passed to us by the parse tables
2    !
2    STATUS = .TPARSE_BLOCK [ TPA$L_PARAM ];
2
2    ! If this is a ambiguity check and there is none return failure
```

```
; 1302        2009  2        ! else signal the error
; 1303        2010  2        !
; 1304        2011  2        IF ( ( .STATUS EQLU FDL$_ABKW ) OR ( .STATUS EQLU FDL$_ABPRIKW ) ) AND
; 1305        2012  3                                          ( NOT .TPARSE_BLOCK [ TPA$V_AMBIG ] )
; 1306        2013  2        THEN
; 1307        2014  2            RETURN 0;
; 1308        2015  2
; 1309        2016  2        ! If this is not a information message the set some error flags
; 1310        2017  2        !
; 1311        2018  3        IF ( NOT ( .CODE [ STS$V_SEVERITY ] EQLU STS$K_INFO ) )
; 1312        2019  2        THEN
; 1313        2020  2
; 1314        2021  2            ! Say that there is an error on this secondary
; 1315        2022  2            !
; 1316        2023  2            FDL$AB_CTRL [ FDL$V_WARNING ] = _SET;
; 1317        2024  2
; 1318        2025  2        ! Signal the error with:
; 1319        2026  2        !
; 1320        2027  2        !     a) Line number
; 1321        2028  2        !     b) Length of the current token
; 1322        2029  2        !     c) Pointer to the token
; 1323        2030  2        !     d) Length of the remainer of the line
; 1324        2031  2        !     e) Pointer to the rest of the line
; 1325        2032  2        !
; 1326        2033  2        SIGNAL ( .TPARSE_BLOCK [ TPA$L_PARAM ],5,
; 1327        2034  2                    .FDL$GL_STMNTNUM,
; 1328        2035  2                    .TPARSE_BLOCK [ TPA$L_TOKENCNT ],
; 1329        2036  2                    .TPARSE_BLOCK [ TPA$L_TOKENPTR ],
; 1330        2037  2                    .TPARSE_BLOCK [ TPA$L_STRINGCNT ],
; 1331        2038  2                    .TPARSE_BLOCK [ TPA$L_STRINGPTR ] );
; 1332        2039  2
; 1333        2040  2        RETURN SS$_NORMAL
; 1334        2041  2
; 1335        2042  1        END;



                                              0000 00000         .ENTRY  FDL$$SYNTAX_ERROR, Save nothing         ; 1953
                  50          20    AC     D0 00002         MOVL    32(TPARSE_BLOCK), STATUS                 ; 2006
       00000000G  8F                50     D1 00006         CMPL    STATUS, #FDL$_ABKW                       ; 2011
                                    09     13 0000D         BEQL    1$                                       ;
       00000000G  8F                50     D1 0000F         CMPL    STATUS, #FDL$_ABPRIKW                    ;
                                    04     12 00016         BNEQ    2$                                       ;
                  2C          06    AC     E9 00018   1$:   BLBC    6(TPARSE_BLOCK), 4$                      ; 2012
  03        50    03                00     ED 0001C   2$:   CMPZV   #0, #3, CODE, #3                         ; 2018
                                    07     13 00021         BEQL    3$                                       ;
       00000000G  00                08     88 00023         BISB2   #8, FDL$AB_CTRL                          ; 2023
                  7E          08    AC     7D 0002A   3$:   MOVQ    8(TPARSE_BLOCK), -(SP)                   ; 2037
                  7E          10    AC     7D 0002E         MOVQ    16(TPARSE_BLOCK), -(SP)                  ; 2035
                      00000000G     00     DD 00032         PUSHL   FDL$GL_STMNTNUM                          ; 2034
                                    05     DD 00038         PUSHL   #5                                       ; 2033
                              20    AC     DD 0003A         PUSHL   32(TPARSE_BLOCK)                         ;
       00000000G  00                07     FB 0003D         CALLS   #7, LIB$SIGNAL                           ;
                  50                01     D0 00044         MOVL    #1, R0                                   ; 2040
                                           04 00047         RET                                              ;
```

```
                                                         50  D4 00048 4$:      CLRL    R0                                               ; 2042
                                                             04 0004A          RET                                                      ;

; Routine Size:  75 bytes,    Routine Base:  _FDL$CODE + 04C8
; 1337            2043  1 %SBTTL  'NEGATE'
; 1338            2044  1 GLOBAL ROUTINE  FDL$$NEGATE : NOVALUE =
; 1339            2045  1 !++
; 1340            2046  1 !
; 1341            2047  1 ! Functional Description:
; 1342            2048  1 !
; 1343            2049  1 !       Produces the negative version of a number
; 1344            2050  1 !
; 1345            2051  1 ! Calling Sequence:
; 1346            2052  1 !
; 1347            2053  1 !       Called from the parse tables
; 1348            2054  1 !
; 1349            2055  1 ! Input Parameters:
; 1350            2056  1 !       none
; 1351            2057  1 !
; 1352            2058  1 ! Implicit Inputs:
; 1353            2059  1 !       none
; 1354            2060  1 !
; 1355            2061  1 ! Output Parameters:
; 1356            2062  1 !       none
; 1357            2063  1 !
; 1358            2064  1 ! Implicit Outputs:
; 1359            2065  1 !       none
; 1360            2066  1 !
; 1361            2067  1 ! Routine Value:
; 1362            2068  1 !
; 1363            2069  1 !       none
; 1364            2070  1 !
; 1365            2071  1 ! Side Effects:
; 1366            2072  1 !       none
; 1367            2073  1 !
; 1368            2074  1 !--
; 1369            2075  1
; 1370            2076  2     BEGIN
; 1371            2077  2
; 1372            2078  2     TPARSE_ARGS;
; 1373            2079  2
; 1374            2080  2     ! Just negate the number
; 1375            2081  2     !
; 1376            2082  2     FDL$GL_NUMBER = -.FDL$GL_NUMBER;
; 1377            2083  2
; 1378            2084  2     RETURN
; 1379            2085  2
; 1380            2086  1     END;



                                      0004 00000          .ENTRY  FDL$$NEGATE, Save R2                                  ; 2044
                     52 00000000G  00   9E 00002          MOVAB   FDL$GL_NUMBER, R2                                     ;
                     62            62   CE 00009          MNEGL   FDL$GL_NUMBER, FDL$GL_NUMBER                          ; 2082
                                        04 0000C          RET                                                           ; 2086

; Routine Size:  13 bytes,    Routine Base:  _FDL$CODE + 0513
; 1382          2087  1 %SBTTL  'SET BLANK'
; 1383          2088  1 GLOBAL ROUTINE  FDL$$SET_BLANK : NOVALUE =
; 1384          2089  1 !++
; 1385          2090  1 !
; 1386          2091  1 ! Functional Description:
; 1387          2092  1 !
; 1388          2093  1 !       Sets the Tparse blanks flag to allow parsing of blanks
; 1389          2094  1 !
; 1390          2095  1 ! Calling Sequence:
; 1391          2096  1 !
; 1392          2097  1 !       Called from the parse tables
; 1393          2098  1 !
; 1394          2099  1 ! Input Parameters:
; 1395          2100  1 !       none
; 1396          2101  1 !
; 1397          2102  1 ! Implicit Inputs:
; 1398          2103  1 !       none
; 1399          2104  1 !
; 1400          2105  1 ! Output Parameters:
; 1401          2106  1 !       none
; 1402          2107  1 !
; 1403          2108  1 ! Implicit Outputs:
; 1404          2109  1 !       none
; 1405          2110  1 !
; 1406          2111  1 ! Routine Value:
; 1407          2112  1 !
; 1408          2113  1 !       none
; 1409          2114  1 !
; 1410          2115  1 ! Side Effects:
; 1411          2116  1 !       none
; 1412          2117  1 !
; 1413          2118  1 !--
; 1414          2119  1
; 1415          2120  2     BEGIN
; 1416          2121  2
; 1417          2122  2     TPARSE_ARGS;
; 1418          2123  2
; 1419          2124  2     ! Just set the flag
; 1420          2125  2     !
; 1421          2126  2     TPARSE_BLOCK [ TPA$V_BLANKS ] = _SET;
; 1422          2127  2
; 1423          2128  2     RETURN
; 1424          2129  2
; 1425          2130  1     END;



                                         0000 00000         .ENTRY  FDL$$SET_BLANK, Save nothing                        ; 2088
                  04   AC             01   88 00002         BISB2   #1, 4(TPARSE_BLOCK)                                 ; 2126
                                           04 00006         RET                                                         ; 2130

; Routine Size:  7 bytes,    Routine Base:  _FDL$CODE + 0520
; 1427        2131   1 %SBTTL  'CLR_BLANK'
; 1428        2132   1 GLOBAL ROUTINE  FDL$$CLR_BLANK : NOVALUE =
; 1429        2133   1 !++
; 1430        2134   1 !
; 1431        2135   1 ! Functional Description:
; 1432        2136   1 !
; 1433        2137   1 !       Clears the Tparse blanks flag
; 1434        2138   1 !
; 1435        2139   1 ! Calling Sequence:
; 1436        2140   1 !
; 1437        2141   1 !       Called from the parse tables
; 1438        2142   1 !
; 1439        2143   1 ! Input Parameters:
; 1440        2144   1 !       none
; 1441        2145   1 !
; 1442        2146   1 ! Implicit Inputs:
; 1443        2147   1 !       none
; 1444        2148   1 !
; 1445        2149   1 ! Output Parameters:
; 1446        2150   1 !       none
; 1447        2151   1 !
; 1448        2152   1 ! Implicit Outputs:
; 1449        2153   1 !       none
; 1450        2154   1 !
; 1451        2155   1 ! Routine Value:
; 1452        2156   1 !
; 1453        2157   1 !       none
; 1454        2158   1 !
; 1455        2159   1 ! Side Effects:
; 1456        2160   1 !       none
; 1457        2161   1 !
; 1458        2162   1 !--
; 1459        2163   1
; 1460        2164   2    BEGIN
; 1461        2165   2
; 1462        2166   2    TPARSE_ARGS;
; 1463        2167   2
; 1464        2168   2     ! Just clear the flag
; 1465        2169   2     !
; 1466        2170   2    TPARSE_BLOCK [ TPA$V_BLANKS ] = _CLEAR;
; 1467        2171   2
; 1468        2172   2    RETURN
; 1469        2173   2
; 1470        2174   1    END;



                                         0000 00000          .ENTRY  FDL$$CLR_BLANK, Save nothing                       ; 2132
                    04   AC           01   8A 00002          BICB2   #1, 4(TPARSE_BLOCK)                                ; 2170
                                           04 00006          RET                                                        ; 2174

; Routine Size:  7 bytes,    Routine Base:  _FDL$CODE + 0527
; 1472           2175  1 %SBTTL  'ERRROR_CHK'
; 1473           2176  1 GLOBAL ROUTINE  FDL$$ERROR_CHK =
; 1474           2177  1 !++
; 1475           2178  1 !
; 1476           2179  1 ! Functional Description:
; 1477           2180  1 !
; 1478           2181  1 !       Does a check if there was a warning
; 1479           2182  1 !
; 1480           2183  1 ! Calling Sequence:
; 1481           2184  1 !
; 1482           2185  1 !       Called from the parse tables
; 1483           2186  1 !
; 1484           2187  1 ! Input Parameters:
; 1485           2188  1 !       none
; 1486           2189  1 !
; 1487           2190  1 ! Implicit Inputs:
; 1488           2191  1 !       none
; 1489           2192  1 !
; 1490           2193  1 ! Output Parameters:
; 1491           2194  1 !       none
; 1492           2195  1 !
; 1493           2196  1 ! Implicit Outputs:
; 1494           2197  1 !       none
; 1495           2198  1 !
; 1496           2199  1 ! Routine Value:
; 1497           2200  1 !
; 1498           2201  1 !       Value of fdl$ab_ctrl [ fdl$v_warning ]
; 1499           2202  1 !
; 1500           2203  1 ! Side Effects:
; 1501           2204  1 !       none
; 1502           2205  1 !
; 1503           2206  1 !--
; 1504           2207  1
; 1505           2208  2     BEGIN
; 1506           2209  2
; 1507           2210  2     TPARSE_ARGS;
; 1508           2211  2
; 1509           2212  2     ! If there is a warning return true else fail
; 1510           2213  2     !
; 1511           2214  2     RETURN .FDL$AB_CTRL [ FDL$V_WARNING ]
; 1512           2215  2
; 1513           2216  1     END;



                                                0000 00000         .ENTRY  FDL$$ERROR_CHK, Save nothing                 ; 2176
          50 00000000G  00          01       03  EF 00002         EXTZV   #3, #1, FDL$AB_CTRL, R0                      ; 2214
                                                  04 0000B         RET                                                  ; 2216

; Routine Size:  12 bytes,    Routine Base:  _FDL$CODE + 052E


; 1514           2217  1
```

```
1 %SBTTL  'FDL$$READ_ERROR'
1 GLOBAL ROUTINE  FDL$$READ_ERROR : NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       This routine will signal an rms error and stop execution if the RMS
1 !       error is NOT end of file.  It is to be used for detecting errors
1 !       during rms $GETs or $READs.
1 !
1 ! Calling Sequence:
1 !
1 !       This routine is call as an AST by RMS
1 !
1 ! Input Parameters:
1 !
1 !       AST argument block which has a pointer to a RAB
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Routines Called:
1 !
1 !       SIGNAL_STOP
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2    BEGIN
2
2    BUILTIN
2       AP;
2
2    BIND
2       AST_BLOCK = AP : REF VECTOR [ ,LONG ];
2
2    LOCAL
2       RAB : REF BLOCK [ ,BYTE ],
2       FAB : REF BLOCK [ ,BYTE ],
2       NAM : REF BLOCK [ ,BYTE ];
2
2    ! Get the rab (Pointer to by the second ast parameter)
2    !
2    RAB = .AST_BLOCK [ 1 ];
2
2    ! If this is only an end of file then return
```

```
; 1573          2275  2         !
; 1574          2276  2         IF .RAB [ RAB$L_STS ] EQLU RMS$_EOF
; 1575          2277  2         THEN
; 1576          2278  2             RETURN;
; 1577          2279  2
; 1578          2280  2         ! Now get the fab it points to
; 1579          2281  2         !
; 1580          2282  2         FAB = .RAB [ RAB$L_FAB ];
; 1581          2283  2
; 1582          2284  2         ! Get the name block
; 1583          2285  2         !
; 1584          2286  2         NAM = .FAB [ FAB$L_NAM ];
; 1585          2287  2
; 1586          2288  2         ! Signal the FDL error with the best file name string
; 1587          2289  2         !
; 1588          2290  2         ! First try the resultant string
; 1589          2291  2         !
; 1590          2292  2         IF .NAM [ NAM$B_RSL ] NEQU 0
; 1591          2293  2         THEN
; 1592          2294  3             BEGIN
; 1593          2295  3             STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_RSL ];
; 1594          2296  3             STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$L_RSA ]
; 1595          2297  3             END
; 1596          2298  3
; 1597          2299  3         ! Next try the expanded string
; 1598          2300  3         !
; 1599          2301  2         ELSE IF .NAM [ NAM$B_ESL ] NEQU 0
; 1600          2302  2         THEN
; 1601          2303  3             BEGIN
; 1602          2304  3             STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_ESL ];
; 1603          2305  3             STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$L_ESA ]
; 1604          2306  3             END
; 1605          2307  3
; 1606          2308  3         ! If all else fails use the name string
; 1607          2309  3         !
; 1608          2310  2         ELSE
; 1609          2311  3             BEGIN
; 1610          2312  3             STRING_DESC [ DSC$W_LENGTH ] = .FAB [ FAB$B_FNS ];
; 1611          2313  3             STRING_DESC [ DSC$A_POINTER ] = .FAB [ FAB$L_FNA ]
; 1612          2314  2             END;
; 1613          2315  2
; 1614          2316  2         SIGNAL_STOP( .RAB [ RAB$L_CTX ],1,STRING_DESC,
; 1615          2317  2                      .RAB [ FAB$L_STS ],.RAB [ FAB$L_STV ] )
; 1616          2318  2
; 1617          2319  1         END;



                                                  000C 00000          .ENTRY  FDL$$READ_ERROR, Save R2,R3                        ; 2219
                        53 00000000'        00  9E 00002          MOVAB   STRING_DESC, R3                                    ;
                        52       04         AC  D0 00009          MOVL    4(AST_BLOCK), RAB                                  ; 2272
               0001827A 8F       08         A2  D1 0000D          CMPL    8(RAB), #98938                                     ; 2276
                                            43  13 00015          BEQL    4$                                                 ;
                        51       3C         A2  D0 00017          MOVL    60(RAB), FAB                                       ; 2282
                        50       28         A1  D0 0001B          MOVL    40(FAB), NAM                                       ; 2286
```

```
                                        03   A0  95 0001F          TSTB    3(NAM)                                    ; 2292
                                             0B  13 00022          BEQL    1$                                        ;
                               63       03   A0  9B 00024          MOVZBW  3(NAM), STRING_DESC                       ; 2295
                       04      A3       04   A0  D0 00028          MOVL    4(NAM), STRING_DESC+4                     ; 2296
                                             19  11 0002D          BRB     3$                                        ;
                                        0B   A0  95 0002F 1$:      TSTB    11(NAM)                                   ; 2301
                                             0B  13 00032          BEQL    2$                                        ;
                               63       0B   A0  9B 00034          MOVZBW  11(NAM), STRING_DESC                      ; 2304
                       04      A3       0C   A0  D0 00038          MOVL    12(NAM), STRING_DESC+4                    ; 2305
                                             09  11 0003D          BRB     3$                                        ;
                               63       34   A1  9B 0003F 2$:      MOVZBW  52(FAB), STRING_DESC                      ; 2312
                       04      A3       2C   A1  D0 00043          MOVL    44(FAB), STRING_DESC+4                    ; 2313
                               7E       08   A2  7D 00048 3$:      MOVQ    8(RAB), -(SP)                             ; 2317
                                             53  DD 0004C          PUSHL   R3                                        ; 2316
                                             01  DD 0004E          PUSHL   #1                                        ;
                                        18   A2  DD 00050          PUSHL   24(RAB)                                   ;
                00000000G      00            05  FB 00053          CALLS   #5, LIB$STOP                              ;
                                                 04 0005A 4$:      RET                                               ; 2319

; Routine Size:  91 bytes,    Routine Base:  _FDL$CODE + 053A


; 1618          2320  1
```

```
%SBTTL  'FDL$$RMS_ERROR'
GLOBAL ROUTINE  FDL$$RMS_ERROR  : NOVALUE =
!++
!
! Functional Description:
!
!       This routine will signal and rms error and stop execution.  It is
!       to be primarly used for detecting errors during asynchronous operations
!
! Calling Sequence:
!
!       This routine is call as an AST by RMS
!
! Input Parameters:
!
!       AST argument block which has a pointer to a rms block
!
! Implicit Inputs:
!       none
!
! Output Parameters:
!       none
!
! Implicit Outputs:
!       none
!
! Routine Value:
!       none
!
! Routines Called:
!
!       SIGNAL_STOP
!
! Side Effects:
!       none
!
!--

    BEGIN

    BUILTIN AP;

    BIND
        AST_BLOCK = AP : REF VECTOR [ ,LONG ];

    LOCAL
        RMS_BLOCK : REF BLOCK [ ,BYTE ];

    ! Get the rms control block (second argument in the block)
    !
    RMS_BLOCK = .AST_BLOCK [ 1 ];

    ! NOTE: We use the RAB$x_zzz codes but they are valid for the FAB as well

    ! Signal the FDL error
    !
    SIGNAL_STOP( .RMS_BLOCK [ RAB$L_CTX ],
; 1677          2378  2                           .RMS_BLOCK [ RAB$L_STS ],.RMS_BLOCK [ RAB$L_STV ] )
; 1678          2379  2
; 1679          2380  1         END;



                                                  0000 00000        .ENTRY  FDL$$RMS_ERROR, Save nothing                    ; 2322
                                 50       04   AC   D0 00002        MOVL    4(AST_BLOCK), RMS_BLOCK                         ; 2371
                                 7E       08   A0   7D 00006        MOVQ    8(RMS_BLOCK), -(SP)                             ; 2378
                                          18   A0   DD 0000A        PUSHL   24(RMS_BLOCK)                                   ; 2377
                         00000000G  00         03   FB 0000D        CALLS   #3, LIB$STOP                                    ;
                                                    04 00014        RET                                                     ; 2380

; Routine Size:  21 bytes,    Routine Base:  _FDL$CODE + 0595


; 1680          2381  1
```

```
1 %SBTTL  'FDL$$RMS_OPEN_ERROR'
1 GLOBAL ROUTINE  FDL$$RMS_OPEN_ERROR : NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       This routine will signal an rms error and stop execution.  It is
1 !       to be primarly used for detecting errors during file opens.
1 !
1 ! Calling Sequence:
1 !
1 !       This routine is call as an AST by RMS
1 !
1 ! Input Parameters:
1 !
1 !       AST argument block which has a pointer to a FAB
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Routines Called:
1 !
1 !       SIGNAL_STOP
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2    BEGIN
2
2    BUILTIN
2        AP;
2
2    BIND
2        AST_BLOCK = AP : REF VECTOR [ ,LONG ];
2
2    LOCAL
2        FAB : REF BLOCK [ ,BYTE ],
2        NAM : REF BLOCK [ ,BYTE ];
2
2    ! Get the fab (Pointer to by the second ast parameter)
2    !
2    FAB = .AST_BLOCK [ 1 ];
2
2    ! If this is really a RAB (from a connect) then get the fab it points to
2    !
2    IF .FAB [ FAB$B_BID ] EQLU RAB$C_BID
```

```
; 1739      2439  2      THEN
; 1740      2440  2          FAB = .FAB [ RAB$L_FAB ];       ! This looks strange but it's ok!
; 1741      2441  2
; 1742      2442  2      ! Get the name block
; 1743      2443  2      !
; 1744      2444  2      NAM = .FAB [ FAB$L_NAM ];
; 1745      2445  2
; 1746      2446  2      ! Signal the FDL error with the best file name string
; 1747      2447  2      !
; 1748      2448  2      ! First try the resultant string
; 1749      2449  2      !
; 1750      2450  2      IF .NAM [ NAM$B_RSL ] NEQU 0
; 1751      2451  2      THEN
; 1752      2452  3          BEGIN
; 1753      2453  3          STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_RSL ];
; 1754      2454  3          STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$L_RSA ]
; 1755      2455  3          END
; 1756      2456  3
; 1757      2457  3      ! Next try the expanded string
; 1758      2458  3      !
; 1759      2459  2      ELSE IF .NAM [ NAM$B_ESL ] NEQU 0
; 1760      2460  2      THEN
; 1761      2461  3          BEGIN
; 1762      2462  3          STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_ESL ];
; 1763      2463  3          STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$L_ESA ]
; 1764      2464  3          END
; 1765      2465  3
; 1766      2466  3      ! If all else fails use the name string
; 1767      2467  3      !
; 1768      2468  2      ELSE
; 1769      2469  3          BEGIN
; 1770      2470  3          STRING_DESC [ DSC$W_LENGTH ] = .FAB [ FAB$B_FNS ];
; 1771      2471  3          STRING_DESC [ DSC$A_POINTER ] = .FAB [ FAB$L_FNA ]
; 1772      2472  2          END;
; 1773      2473  2
; 1774      2474  2      SIGNAL_STOP( .FAB [ RAB$L_CTX ],1,STRING_DESC,
; 1775      2475  2                  .FAB [ FAB$L_STS ],.FAB [ FAB$L_STV ] )
; 1776      2476  2
; 1777      2477  1      END;



                            0004 00000          .ENTRY   FDL$$RMS_OPEN_ERROR, Save R2                ; 2383
      52 00000000'    00  9E 00002          MOVAB    STRING_DESC, R2                             ;
      51        04    AC  D0 00009          MOVL     4(AST_BLOCK), FAB                           ; 2434
      01              61  91 0000D          CMPB     (FAB), #1                                   ; 2438
                      04  12 00010          BNEQ     1$                                          ;
      51        3C    A1  D0 00012          MOVL     60(FAB), FAB                                ; 2440
      50        28    A1  D0 00016 1$:      MOVL     40(FAB), NAM                                ; 2444
                03    A0  95 0001A          TSTB     3(NAM)                                      ; 2450
                      0B  13 0001D          BEQL     2$                                          ;
      62        03    A0  9B 0001F          MOVZBW   3(NAM), STRING_DESC                         ; 2453
 04   A2        04    A0  D0 00023          MOVL     4(NAM), STRING_DESC+4                       ; 2454
                      19  11 00028          BRB      4$                                          ;
                0B    A0  95 0002A 2$:      TSTB     11(NAM)                                     ; 2459
```

```
                                               0B  13 0002D           BEQL    3$                              ;
                                 62       0B   A0  9B 0002F           MOVZBW  11(NAM), STRING_DESC            ; 2462
                          04     A2       0C   A0  D0 00033           MOVL    12(NAM), STRING_DESC+4          ; 2463
                                               09  11 00038           BRB     4$                              ;
                                 62       34   A1  9B 0003A  3$:      MOVZBW  52(FAB), STRING_DESC            ; 2470
                          04     A2       2C   A1  D0 0003E           MOVL    44(FAB), STRING_DESC+4          ; 2471
                                 7E       08   A1  7D 00043  4$:      MOVQ    8(FAB), -(SP)                   ; 2475
                                               52  DD 00047           PUSHL   R2                              ; 2474
                                               01  DD 00049           PUSHL   #1                              ;
                                          18   A1  DD 0004B           PUSHL   24(FAB)                         ;
                     00000000G   00            05  FB 0004E           CALLS   #5, LIB$STOP                    ;
                                               04  00055              RET                                     ; 2477

; Routine Size:  86 bytes,    Routine Base:  _FDL$CODE + 05AA


; 1778          2478  1
; 1779          2479  0 END ELUDOM




                                                                      .EXTRN  LIB$SIGNAL, LIB$STOP

;                                    PSECT SUMMARY
;
;       Name                        Bytes                      Attributes
;
;  _FDL$OWN                            12  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _FDL$CODE                         1536  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)




;                              Library Statistics
;
;                                           -------- Symbols --------        Pages        Processing
;       File                                Total    Loaded   Percent        Mapped       Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1         9776        39         0          581         00:01.0





;                                    COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:FDLDRIVER/OBJ=OBJ$:FDLDRIVER MSRC$:FDLDRIVER/UPDATE=(ENH$:FDLDRIVER)

; Size:           1536 code + 12 data bytes
; Run Time:          00:36.1
; Elapsed Time:      02:08.7
; Lines/CPU Min:       4125
; Lexemes/CPU-Min:  21518
; Memory Used:  175 pages
; Compilation Complete
```

0176 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

FDLDATA
LIS

FDLUTIL
REQ

FDLSHR
MAP

FDLPARDEF
SDL

FDLCALL
LIS

CREATEFDL
LIS

FDLDRIVER
LIS

FDLGEN
LIS